

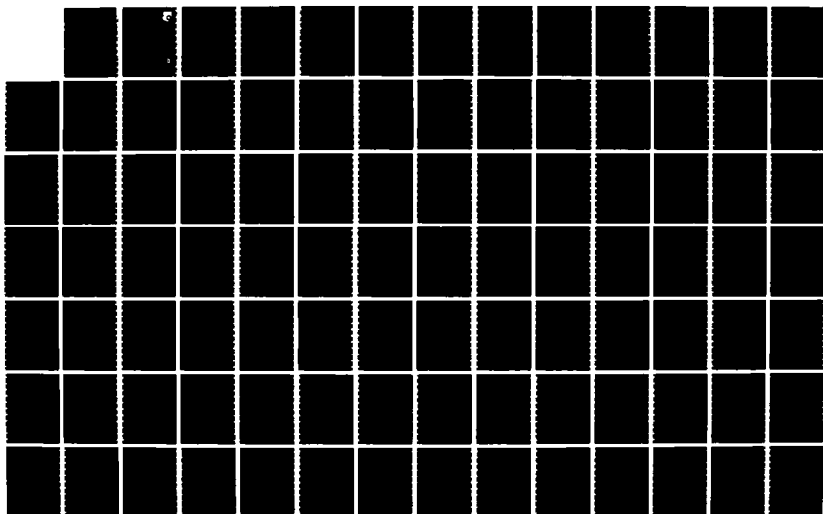
AD-A173 028

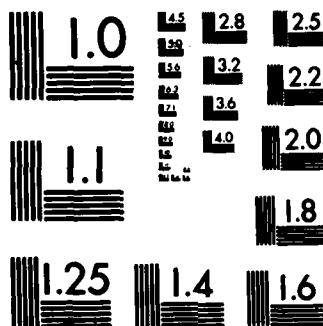
INFORMATION PROCESSING RESEARCH(U) CARNEGIE-MELLON UNIV 1/2
PITTSBURGH PA DEPT OF COMPUTER SCIENCE E BALL ET AL
SEP 86 AFMAL-TR-86-1011 F33615-81-K-1539

UNCLASSIFIED

FFG 9/2

ML





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A173 028

2

AFWAL-TR-86-1011



INFORMATION PROCESSING RESEARCH

Ball, Bentley, Haberman, Hibbard, Kanade, Kung, McDermott, Newell, Rashid,
Reddy, Robertson, Sproull, and Wulf
The Carnegie-Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

September 1986

Final Report for Period 1 January 1981 to 31 December 1984

Approved for public release; distribution unlimited.

DTIC FILE COPY

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6503

DTIC
ELECTE
OCT 16 1986
S D E


86 10 16 085

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



CHAHIRA M. HOPPER
Project Engineer



RONALD L. RINGO, Acting Chief
Information Processing
Technology Branch
System Avionics Division

FOR THE COMMANDER



MARVIN SPECTOR, Acting Chief
System Avionics Division
Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAT-3 W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified

ADA173028

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY -			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release, distribution unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE -			5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-86-1011		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) -			7a. NAME OF MONITORING ORGANIZATION AFWAL/AAAT-3		
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University		6b. OFFICE SYMBOL (If applicable) CMU		7b. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB Dayton, Ohio 45433-6543	
6c. ADDRESS (City, State and ZIP Code) 5000 Forbes Avenue Pittsburgh, Pennsylvania 15213			7c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB Dayton, Ohio 45433-6543		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION DARPA		8b. OFFICE SYMBOL (If applicable) -		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-81-K-1539	
8c. ADDRESS (City, State and ZIP Code) 1400 Wilson Boulevard Arlington, VA 22209			10. SOURCE OF FUNDING NOS.		
11. TITLE (Include Security Classification) Information Processing Research			PROGRAM ELEMENT NO. 61101E	PROJECT NO. 3597	TASK NO. 00
					WORK UNIT NO. 02
12. PERSONAL AUTHOR(S) Ball, Bentley, Haberman, Hayes, Hibbard, Kanade, Kung, McDermott, Newell, Rashid, Reddy, Wulf					
13a. TYPE OF REPORT FINAL		13b. TIME COVERED FROM Jan 81 to Dec 84		14. DATE OF REPORT (Yr., Mo., Day) 86/Sep	
				15. PAGE COUNT 149	
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD 05	GROUP 08	SUB. GR.	Distributed Processing, Image Understanding, Machine Intelligence, Distributed Sensor Network, Cooperative User Interface, Integrated VLSI Systems, Natural Language, Shadow Geometry, Gradient Space, Code Optimization Compiling Techniques,		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>→ This report documents DARPA supported basic research in Carnegie-Mellon University's Computer Science Department during the period 1 January 1981 through 31 December 1983, extended to 31 December 1984. Each chapter discusses one of seven major research areas. Sections within a chapter present the area's general context, the specific problems addressed, our contributions and their significance, and an annotated bibliography.</p>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Chahira Hopper			22b. TELEPHONE NUMBER (Include Area Code) (513) 255-7865		22c. OFFICE SYMBOL AFWAL/AAAT-3

18. (Cont.)

Parallel Architectures Network Interprocess Communication, Dynamic Load Balancing Flexible Parsing, Integrated Speech, Natural Language Algorithm Design Theory.

19. (Continued)

The research areas and their main objectives are:

- *Distributed Processing:* Develop techniques and systems for effective use of computer networks. This effort involves developing a methodology for efficient utilization of distributed (loosely connected) personal computers.
- *Image Understanding:* Apply knowledge effectively in assisting the image interpretation process. This work deals with systems that perceive the external world through visual images, extract useful information, and pass such information to another system that then employs it to accomplish some larger task.
- *Machine Intelligence:* Investigate ways to utilize knowledge in obtaining intelligent action by computers. Long range goals of this effort include the discovery of principles which enable intelligent action and the construction of computer systems which can perform tasks requiring intelligence.
- *Programming Technology:* Increase our ability to produce predictably high-quality software systems.
Explore and evaluate alternative techniques for the effective use of very large memories. This effort focuses on the design and construction of a memory hierarchy incorporating video disk technology.
- *Distributed Sensor Networks:* Construct a demonstration system of physically and logically distributed computers interacting through a communication network to identify, track, and display the situation of multiple objects in a laboratory environment.
- *Cooperative User Interfaces:* Investigate methods for increasing human productivity by improving the efficiency and effectiveness of man-machine communication. These techniques will be incorporated in a user interface system and evaluated in the department's research computing environment.
- *Integrated VLSI Systems:* Advance our ability to design and apply in real systems the high-density digital circuits possible through emerging VLSI technologies.

Table of Contents

1. Introduction	1-1
1.1. Scope of Research	1-1
1.2. The CMU Research Environment	1-3
2. Research in Distributed Processing	2-1
2.1. Programming Languages and Environments	2-2
2.2. Programs and Facilities	2-3
2.2.1. Application programs	2-3
2.2.2. User Interface Systems	2-3
2.2.3. Distributed Resource Management	2-5
2.2.4. Maintenance Programs	2-6
2.3. The Distributed File System	2-6
2.3.1. Prototype storage systems	2-6
2.3.2. Methodology for evaluating storage systems	2-7
2.4. Bibliography	2-8
3. Research in Image Understanding	3-1
3.1. Understanding Three-Dimensional Images	3-1
3.1.1. Gradient space theory	3-1
3.1.2. Shadow geometry theory	3-2
3.1.3. Generalized cylinders	3-2
3.1.4. Image matching by two-stage dynamic programming	3-3
3.2. Modeling a Three-dimensional World	3-4
3.2.1. Modeling from aerial photography	3-5
3.2.2. Modeling from 3-D Range Imagery	3-5
3.3. Applications in Digital Mapping and Photointerpretation	3-6
3.3.1. An integrated spatial database	3-6
3.4. Vision for Navigation	3-7
3.4.1. A visual navigation system for the CMU rover	3-8
3.4.2. Obtaining camera motions from image sequences	3-8
3.4.3. Obtaining object motion from image sequences	3-8
3.5. Contributions to the SRI Testbed Facilities	3-9
3.6. Bibliography	3-10
4. Machine Intelligence	4-1
4.1. Knowledge Representation	4-1
4.1.1. Machine Learning	4-2
4.1.2. Adversarial Problem Solving	4-2
4.1.3. Massively Parallel Cognitive Architectures	4-4
4.1.4. A Universal Problem-Solving Architecture	4-5
4.2. Knowledge Engineering	4-6
4.2.1. An Algorithm Design Assistant	4-6
4.2.2. Architectures for Fast, Intelligent Search	4-7
4.2.3. Massive Search Systems	4-7
4.2.4. Prodigy: A Learning Apprentice	4-8
4.2.5. Speech Recognition	4-9



Accession For	
NTIS	GRA&I <input checked="" type="checkbox"/>
DTIC	TAB <input type="checkbox"/>
Unannounced <input type="checkbox"/>	
Justification _____	
By _____	
Distribution/ _____	
Availability Codes _____	
Dist	Avail and/or Special
A-1	

TABLE OF CONTENTS

4.2.6. SPAM: Rule-Based Systems for Aerial Photointerpretation	4-10
4.3. Bibliography	4-12
5. Research in Programming Technology	5-1
5.1. Automating Compiler Construction	5-1
5.1.1. Analyzing the target language	5-1
5.1.2. Creating "generating-systems" techniques	5-2
5.1.3. Finding code optimizing compiling techniques	5-3
5.2. Highly Secure and Reliable Systems	5-3
5.2.1. Design and implementation of parallel architectures and software	5-4
5.2.2. Verification concepts, techniques, and applications on parallel architectures	5-5
5.2.3. Specific design of fault tolerance and reliability support in parallel systems	5-5
5.3. Advanced Programming Environments	5-6
5.3.1. Designing a programming environment	5-6
5.3.2. Automatically generating environments	5-7
5.4. Bibliography	5-9
6. Research in Distributed Sensor Networks	6-1
6.1. A Communication-Oriented Operating System	6-1
6.2. Protocols for Network Interprocess Communication	6-2
6.3. An Interface Specification Language	6-3
6.4. Dynamic Load Balancing and Fault Reconfiguration	6-3
6.5. Prototype for Distributed Sensing	6-4
6.6. Transaction Based Systems	6-4
6.7. Bibliography	6-5
7. Research in Cooperative User Interfaces	7-1
7.1. Robust Communication	7-2
7.2. Robust, Flexible Parsing	7-2
7.3. Cooperative Error-correction	7-3
7.4. Media-rich Communication	7-5
7.5. Integrating Speech and Natural Language	7-5
7.6. Explanation	7-6
7.7. Bibliography	7-7
8. Research in Integrated VLSI Systems	8-1
8.1. VLSI Theory	8-1
8.1.1. Algorithms	8-2
8.1.2. Theory of algorithm design	8-2
8.1.3. Implementation issues	8-3
8.2. VLSI Systems	8-3
8.2.1. Architectures	8-3
8.2.2. Chips	8-5
8.3. VLSI Design Tools	8-6
8.4. Bibliography	8-8
Index	-1

1. Introduction

This report documents basic computer science research in Carnegie Mellon University's Computer Science Department. The Information Processing Techniques Office of the Defense Advanced Research Projects Agency (DARPA) supported this work during the period 1 January 1981 through 31 December 1983, extended to 31 December 1984.

The remainder of this chapter describes our research scope and the CMU-CSD research environment. Chapters 2 through 8 then present in detail our seven major research areas: Distributed Processing, Image Understanding, Machine Intelligence, Programming Technology, Distributed Sensor Networks, Cooperative User Interfaces, and VLSI Systems. Sections in each chapter present the area's general research context, the specific problems we addressed, our contributions and their significance, and an annotated bibliography. The index provides access via keywords and system names.

The bibliographies present selected references that reflect the scope and significance of CMU's contributions to basic computer science research. Wherever possible, particularly for key articles, we have included abstracts. While we have striven for comprehensive coverage, some documents have regrettably eluded our efforts. Finally, though basic research does not proceed with the mechanical regularity of industrial production, publication dates do indicate progress in the various problem areas. CSD Technical Report dates exhibit the closest correlation with temporal progress and the report text frequently reappears later in the more accessible archival literature.

1.1. Scope of Research

We organize the research reported here under seven major headings. These interrelated categories and their major objectives are:

- *Distributed Processing (DP)*: Develop techniques and systems for effectively using computer networks. This effort involves developing a methodology for efficiently utilizing distributed (loosely connected) personal computers. Research on a concept demonstration system proceeds in several areas:
 - Design and implementation of basic system software facilities including an operating system kernel, an interprocess communication facility, and a high-level development environment.
 - Integration of subsystems and services at two levels—the user interface and the underlying system architecture—in order to provide significant improvement in the productivity of computer science researchers.
 - Design and implementation of two programming systems to support a variety of applications.

- Development of a distributed file system, one of the principal centralized services that support both the network of personal computers and larger, time-shared systems.
- Building an interactive document preparation system by merging existing packages into an integrated environment.
- Extension of current message systems to handle multimedia formats by exploiting the technology of personal computers and their interconnecting network.

Explore and evaluate alternative techniques for effectively using very large memories. Long-term goals of this effort focus on designing and constructing a memory hierarchy incorporating video disk technology. Research in archival memory systems involves the following sub-tasks:

- Build the network-based Central File System.
 - Integrate the Central File System into an environment of several heterogeneous computer systems.
 - Investigate data organization and data management strategies for storage media with characteristics similar to those of video disks.
 - Construct a simulation-based tool for design and analysis of memory hierarchies.
- *Image Understanding (IU):* Apply knowledge effectively in assisting the image interpretation process. This work deals with systems that perceive the external world through visual images, extract useful information, and pass such information to another system that then employs it to accomplish some larger task. Research in this area aims at:
 - Understanding and constructing systems which can comprehend three-dimensional structure in the environment from a two-dimensional visual image.
 - Discovering the representations, algorithms, and control structures required to exploit pre-existing knowledge about the environment for image understanding.
 - Inventing special architectures and programming structures to realize the algorithms efficiently.
 - *Machine Intelligence (MI):* Investigate ways to utilize knowledge in obtaining intelligent action by computers. Long range goals of this effort include the discovery of principles which enable intelligent action and the construction of computer systems which can perform tasks requiring intelligence. Research in machine intelligence covers a wide range of issues:
 - Discovering and analyzing methods of problem solving.
 - Discovering and analyzing the ways problems may be represented and how such representations affect the difficulty of solving the problems.
 - Discovering and analyzing processes which produce appropriate internal representations through recognition and description of external task situations.
 - Discovering and understanding control structures and system organizations which can combine a collection of problem-solving methods and problem representations into an effective total system.
 - *Programming Technology (PT):* Increase our ability to produce predictably high-quality software systems. Research in this area strives to:
 - Automate the construction of compilers.

INTRODUCTION

- Develop a highly secure and reliable system.
- Develop advanced programming environments that facilitate tool integration, system version maintenance, and project management.
- Conduct basic research in programming technology that is likely to lead to additional techniques for producing high-quality systems.
- *Distributed Sensor Networks:* Construct a demonstration system of physically and logically distributed computers interacting through a communication network to identify, track, and display the situation of multiple objects in a laboratory environment. This project will involve the following tasks:
 - Evaluate the design and performance of our current Testbed system.
 - Extend the Testbed through the addition of capabilities for motion perception, visual sensing, multi-object tracking, and multi-sensor integration.
 - Investigate design and implementation issues basic to distributed computing: architecture, language primitives, and descriptive representation.
- *Cooperative User Interfaces:* Investigate methods for increasing human productivity by improving the efficiency and effectiveness of man-machine communication. These techniques will be incorporated in a user interface system and evaluated in the department's research computing environment. The effort will concentrate on the following goals:
 - Shift the design emphasis from convenience of the system builder to that of the system user.
 - Employ a more cooperative style of user interaction, including the ability to negotiate with the user to correct a misunderstanding.
 - Apply newly available hardware capabilities to expand and enrich the mechanisms of communication between man and machine.
- *Integrated VLSI Systems:* Advance our ability to design and apply in real systems the high-density digital circuits possible through emerging VLSI technologies. Achieving this objective relies on several interrelated efforts:
 - Shift theoretical attention from computational complexity to developing applied VLSI algorithms for both chips and programmable arrays.
 - Build a system of integrated design tools sufficiently coordinated so that designers can effectively and routinely carry out their tasks.
 - Produce, using these strategies, a VLSI chip that is an integral part of an operational system and evaluate the performance of that system against realistic criteria.

1.2. The CMU Research Environment

DARPA-supported computer science research in the Carnegie Mellon environment tends to focus around specific experimental systems that strive toward particular objectives, for example, developing a distributed processor system or demonstrating an image understanding system. This report describes approximately two dozen such activities. Sometimes creating and demonstrating a system will itself represent an appropriate

scientific objective. At other times, some particular performance level constitutes our goal. Often, however, the system merely provides a vehicle that permits exploring and investigating basic scientific questions. Thus our work tends to emphasize concept demonstration rather than system engineering. In short, though they don't always represent ends in themselves, our research systems form a convenient structure for organizing and discussing DARPA projects at CMU.

A major strength of the Carnegie Mellon University environment is the synergy resulting from the close cooperation and interdependence of various research efforts despite their diverse foci. For example, AI efforts have often needed the benefits of novel computer architecture and software techniques. Conversely, techniques developed in AI have been used to solve some of the combinatorial problems arising in compiler design and circuit layout. Close interaction and cooperation between the various research efforts has led to new and innovative approaches and solutions, and has significantly contributed to the intellectual ferment that makes Carnegie Mellon University unique in the computer science area.

We have no administrative structure corresponding to our effort organization. We consist simply of faculty, research scientists, and graduate students of the Computer Science Department, with facility support divided between an Engineering Laboratory and a Programming Group. The remaining structure is informal. This organizational style minimizes barriers between efforts and promotes the interactions and synergy reflected in the work distribution shown in Table 1-1.

	Number of Areas	DP	IU	MI	PT	DSN	CUI	VLSI
• Eugene Ball	2	x	□
• Jon Bentley	1	□
Mario Barbacci	1	x
Hans Berliner	1	.	.	x
Roberto Bisiani	2	.	.	x	.	.	.	x
Scott Fahlman	2	x	.	x
Allan Fisher	1	x
Charles Forgy	1	.	.	x
• Nico Habermann	1	.	.	.	□	.	.	.
Samuel Harbison	1	x
Phil Hayes	1	□	.
• Peter Hibbard	2	□	.	.	.	x	.	.
Paul Hilfinger	1	.	.	.	x	.	.	.
Anita Jones	1	.	.	.	x	.	.	.
• Takeo Kanade	1	.	□
Elaine Kant	1	.	.	x
John Kender	1	.	x
H. T. Kung	1	□
John McDermott	1	.	.	□
Hans Moravec	1	.	x
John Nestor	1	.	.	.	x	.	.	.
Joe Newcomer	1	.	.	.	x	.	.	.
Allen Newell	1	.	.	□
Richard Rashid	2	x	.	.	.	□	.	.
Raj Reddy	4	.	□	x	.	□	x	.
George Robertson	1	□
Mike Rychener	2	.	.	x	.	.	x	.
William Scherlis	1	.	.	.	x	.	.	.
Mary Shaw	1	.	.	.	x	.	.	.
Herb Simon	1	.	.	x
Bob Sproull	1	□
Guy Steele	1	x
Howard Wactlar	1	x
Bill Wulf	1	.	.	.	□	.	.	.

x = Active research in this area

□ = Responsible for area

Table 1-1: Faculty effort distribution, 1981-84

2. Research in Distributed Processing

Advances in network technologies and the increasingly sophisticated requirements of users present us with the challenge of developing more effective computing facilities. Traditionally, resources have been provided by timeshared systems. High-performance personal computers—powerful, single-user machines providing quality graphics support such as a bit-map screen, a pointing device, and good networking capabilities—offer substantial benefits over timeshared systems. Their consistently high availability of computing cycles and high bandwidth at the man-machine interface allow a user-to-software interaction level that timeshared environments cannot match. Timeshared systems do, however, allow extensive communication among users and impose a coherent set of standards on the tools they provide. A shift away from a timeshared environment to a personal computing environment must retain these features.

Our research goal in Distributed Processing was to exploit networked personal computers effectively by developing techniques and systems that will:

- Support large programs
- Provide simultaneous multiprogramming capabilities
- Offer language-independence
- Enforce a high degree of protection
- Exploit the hardware's unique features (e.g. graphics capabilities)
- Be easily portable to other architectures

To attain our goal Distributed Processing researchers worked closely with other project scientists and developed Spice, the Scientific Personal Integrated Computing Environment. Spice includes a complete software system, over 160 scientific personal computers, and a packet switching network providing high-bandwidth interprocess communication.

Distributed Sensor Net researchers developed the Spice operating system kernel, Accent (see also Section 6.1) [Rashid 84]. During this contract period we also implemented prototypes for several other projects that utilize Spice services, though we do not support them directly. They include: the Gandalf Aloe editor-generator system, the Descartes user interface management system, the DP Drawing Program, the Spoonix Unix simulator, the Matchmaker remote procedure call generator, and the Cousin Cooperative User Interface system (see Chapter 7). Our efforts can be broken down into the following categories.

- Programming languages and environments
- Programs and facilities
- Archival memory: the distributed file system

2.1. Programming Languages and Environments

To simplify transporting Spice to other personal machines, we wanted to make Spice language independent and provide it with interlanguage communication facilities. We achieved this by supplementing the Perq's manufacturer-supplied Pascal tools with comprehensive programming environments for Ada and Lisp, each with its own microcode interpreter.

We chose a subset of Ada syntax and semantics for Spice that allowed us to use a modified version of the Perq Pascal compiler. Since the compiler generates the same object code as the Pascal compiler, subset Ada and Pascal are link level compatible. Ada researchers worked on designing and building: an incremental Ada compiler, a source-level debugger called Kraut, a run-time representation for Ada programs suitable for interactive programming environments, and various program management tools [Hibbard 81].

No existing Lisp could be ported directly to Spice because of Perq microcode limitations. Thus we chose to develop Common Lisp (a consolidation of Lisp development efforts) [Steele 84] and from that, Spice Lisp. Our design of Common Lisp and implementation of Spice Lisp is now complete [Fahlman 84].

One of the important problems encountered in implementing large distributed systems is debugging their code. In addition to traditional process-level debugging, developers need to know the system's state, understand system component interconnections, and monitor communication among processes. During the contract period, we built two debugging facilities addressing these issues: BlackFlag and Kraut.

We developed BlackFlag as a display-oriented debugger for the DSN testbed system (discussed in Chapter 6). Built as a collection of cooperating processes, BlackFlag facilitated several important debugging operations:

- It provided a graphical display of the system's process communication structure.
- It allowed the kernel to intercept, monitor, and even modify messages before passing them on to their destination processes.
- It provided for source-level debugging of individual processes.

The Kraut debugger continued, in many respects, the BlackFlag effort. While BlackFlag was largely a prototype system that demonstrated the ideas behind distributed debugging, Kraut evolved to be the production debugger for the Accent environment— supporting source-level debugging of both Pascal and Ada programs. Kraut provides most of the commands of traditional symbolic debuggers, such as setting of breakpoints, state inspection/modification and source file access. It also contains low-level debugger commands for inspection of the target process at the code and microcode level. A novel feature of Kraut is its use of Path-Rules as a mechanism for describing conditions to monitor and test on running programs. It allows

the use of graphics to represent the state of variables and other information about a program. We implemented Kraut using a rule-based paradigm that provided substantial flexibility and user tailorability. We also began work, to be reported on later, on solutions to problems related to the portability of such debugging features across machines and for different languages.

2.2. Programs and Facilities

To make Spice a feasible alternative to timeshared systems, we developed numerous application systems, including a text editor, a document formatter, a graphics package, and a mail system. Our user interface package, distributed resource manager, and maintenance programs help make Spice a habitable working environment. Most Spice programs follow system-wide conventions to answer simple help requests and the Spice Documentation group has produced a variety of user documents, including an introductory guide, a guide to system utilities, and manuals for all major components.

2.2.1. Application programs

Spice offers several programs to handle text. Early in the project we built an emacs-like editor, Oil [Wright 84], based on Pepper, an editor that ran on Perqs before Accent was implemented. We later developed Mint, a Scribe-like document formatter [Anderson 84].

We designed and wrote the Spice mail system, Mercury, in Ada. It provides flexibility and power similar to the RdMail facility developed at CMU for the PDP-10 (Tops-10). Mercury allows users to read, answer, store, retrieve, and organize their electronic mail. We originally brought Mercury up with the Subada compiler on a VAX and then ported it, along with the SubAda compiler to Perqs in spring of 1984. A mail delivery mechanism based on transparent Accent message passing to remote VAXes (where mailboxes were maintained) was implemented to support the Perq implementation of Mercury.

2.2.2. User Interface Systems

Canvas [Ball 81], the Spice graphics utility, provides two levels of abstraction for modeling the physical screen. The first is the canvas, which is a region having a particular user-defined coordinate space. Graphics operations--painting with a color, drawing lines, selecting and using a particular font--take place in terms of the canvas coordinates. These operations correspond to the raster operations that would be performed on the physical screen were the program to have direct access to it. The second level of abstraction provides for a canvas that is not necessarily visible. The visibility and the position on the screen of pixels that have been written into a canvas are determined by viewport and the associated refresh tree.

We devised various systems for the creation and support of user interfaces within the Spice environment.

The multiplexing of screen, keyboard, and pointing device among several processes was originally performed by Canvas, a separate process that had a part of its virtual address space mapped to the physical memory used for the screen's bit-map. A user interface package called Environment supplied the means of invoking programs, and the mechanism for providing them with the environment they needed during execution. In particular it provided programs with parameters, switches and commands, and means for presenting information on the screen.

Sapphire is the successor to Canvas/Environment as the window managing system for the Spice environment. Sapphire supports a full implementation of the covered window paradigm (where the rectangular windows can overlap like pieces of paper on a desk). Windows can cover each other, can extend off the screen in any direction, or may lie entirely offscreen. Windows in Sapphire usually have title lines and borders. Application programs may create windows without either, but the borders are useful for showing where the windows are, and the title lines are useful for displaying status information. In Spice, the title line might contain the current working directory. A window running a compiler might have the version number of the compiler and the name of the file being processed displayed in the title line. One of Sapphire's goals is to provide a rich and powerful user interface without restricting the user interface of applications running under it. This is important, since the user will be giving commands to the application program far more often than to the window manager. Sapphire can be used to support many different types of applications with different input and output requirements. Sapphire uses icons to enhance the user's productivity when executing multiple tasks concurrently. Users will often to have several tasks performed simultaneously to increase their efficiency. However, people easily lose track of what they are doing and need aids to help plan, monitor, and control the various tasks operating at the same time. The icons in Sapphire present six kinds of information about the process being run, as well as two kinds of information about the status of the window:

- Process name
- Linear progress bar, showing approximate progress as a percentage of the entire job
- Random progress bar, showing by a constantly changing pattern that progress is being made, though without indicating "how much"
- Error status
- Waiting for user input
- Application defined attention signal
- Listener status, by highlighting the border of the icon of the Listener window
- Offscreen status, by showing three dots (...) to indicate a window is no longer visible

2.2.3. Distributed Resource Management

The sharing of network resources is complicated by issues of security and autonomy, since a network of personal computers may be composed of nodes that are completely controlled by their owners. A network of personal computers has the characteristic that its resources are distributed. In spite of the advantages of distributed resources, a network of personal computers also has some disadvantages. For example, a user may need to access data that is only available on a remote machine. Security may dictate that the data cannot be transferred in whole to any other machine; thus, the user must use a remote processor to access the data. Another disadvantage is that the physical distribution of resources may not match the distribution of the demands for service. Thus, some resources may be idle while others are overloaded. Finally, even though a personal computer may have significant computational capabilities, its power is less than that expected of a large mainframe computer. As a consequence, a network may collectively have tremendous computing power, but its computing resources are distributed. Programs that might be practical on a time-shared mainframe computer may be inappropriate for personal computers because of the amount of computation involved. All of these problems can be alleviated by resource sharing.

To facilitate sharing in this sort of environment, an operating system component called the Butler was proposed [Dannenberg 82]. As a host, the Butler is responsible for administering a sharing policy on its local machine. This includes authenticating sharers, granting rights in accordance with a locally established policy, and creating execution environments for guests. As an agent, the Butler negotiates with hosts on remote machines to obtain resources requested by a client, and performs authentication to discourage a remote host from exploiting the client.

To protect a machine from exploitation by a guest process, the Butler relies upon a capability-based accounting system called the Banker, which keeps track of resource utilization by guests, and provides mechanisms for revoking service. Accounting offers a solution to the problem of laundered requests, where a guest performs malicious operations through a privileged intermediary. The Banker's revocation mechanism is useful in notifying all of a guest's servers that the guest's privileges have been reduced.

Although negotiation is designed to reduce the probability of revocation, a hierarchical recovery scheme is supported by the Butler as an aid to the application programmer in cases where revocation does occur. The three recovery methods are warning, where the guest is allowed to perform application specific actions to free resources, deportation, where the guest is transported to another site by Butlers, and termination, where the guest is simply aborted.

We implemented a Butler prototype that, though it provides only partial functionality of the Butler, does support a real distributed processing application. We limited the implementation to remote invocation and

deportation, because these are areas where performance is an important factor, and because these areas seemed feasible to investigate, given the state of the implementation of the Spice system. In each case, we instrumented the prototype to measure the operation cost in terms of actual processing time, and also in more abstract terms to achieve some degree of technology independence in the results.

2.2.4. Maintenance Programs

Update [Giuse 84], a system built to support the distributed archival and retrieval of system files, has been used as the chief distribution and archival mechanism since 1983 and it is currently used to handle all the data storage needs of the Spice environment. Our primary goal in designing the Update system was to provide a simple and uniform way for workstation users to retrieve all or parts of a software system. Update was also meant to provide an automatic change-log facility that would allow maintainers to record information about system changes at the time the system was being released. Another important goal was to minimize network traffic involved in a transfer, given the number of workstations and the potential for simultaneous requests for any given set of archived files. As a result, the design of Update incorporated the ability to retrieve selectively only files that are different in a new release. Files that haven't changed should never be transferred across the network. Another design goal was to minimize file system usage on the remote machines. To achieve this goal, Update stores only files that have changed. Files that are common to two or more versions are not copied; a link is made instead, using the UNIX file-link mechanism. This approach results in considerable space savings.

2.3. The Distributed File System

Under the Archival Memory project, CMU researchers developed a centralized file system (CFS) to provide secure, reliable storage and archiving facilities for files from all departmental computers. We developed a similar storage system for Spice along with a methodology for evaluating storage systems. Originally we had planned to integrate video disks into our file system archive servers. However, commercial manufacturers have yet to release inexpensive, high-quality, write-once video disks. Thus we merged the Archival Memory project with our Distributed Processing work.

2.3.1. Prototype storage systems

Our design for the Spice File System incorporated a subset of the earlier Central File System (CFS) design and was intended for initial installation on the Perq computers. The full specification was to appear later on unspecified Central Server Machines. We devoted several months in early 1982 to building an interface for the Perq file system. The interface implemented a subset of calls specified by the Spice/CFS File System design. This effort was essentially meant to provide a compatibility package for the existing file system, thus

allowing incremental conversion. Towards the end of that period it became clear that the design, worked out two years earlier, did not easily meet all the requirements of a distributed environment. We then undertook a redesign effort, producing the current design for Sesame, as the file system is now known [Thompson 85]. Significant changes included moving all protection issues into the name space, and allowing only invariant files. We also now expect that Central Server Machines will be standard Spice Machines, with larger disks, and eventually archival media, but running essentially the same software as a personal Spice Machine. This uniformity should provide many advantages over a scheme drawing a hard distinction between user and server machines.

In addition to being a file storage service, Sesame provides most of the interrelated services needed to allow protected sharing of data and services in a network of personal and central computers. It deals with user verification issues both locally and between machines, name lookup services for various typed objects, archiving of files to more stable media as well as the fundamental functions of reading and writing files. Sesame is currently running as an alternate file system in the Spice environment. Each service is independently implementable on other hosts (e.g. UNIX VAXes) on the local net. Researchers are now testing and debugging the system.

2.3.2. Methodology for evaluating storage systems

We developed a methodology for modeling storage devices and subsystems. Our strategy separates device-dependent from hierarchy-dependent characteristics, thereby permitting us to use off-the-shelf software in simulations of memory hierarchy performance. To demonstrate the methodology's effectiveness, we used it to build a simulation tool that runs under UNIX and applied it to the CMU network file system [Satyanarayanan 81, Satyanarayanan 83]. For design purposes, this modeling tool takes partial specifications of an architecture and produces ranges of complete specifications including: environmental parameters (e.g. load, reference patterns, capacity requirements), software parameters (e.g. migration strategies), and hardware parameters (e.g. cost, access time, bandwidth, capacity, media lifetime). For analysis purposes, the tool takes complete specifications, and helps diagnose problems (e.g. improperly set migration strategies).

2.4. Bibliography

- [Anderson 84] Anderson, P., P. Hibbard, and K. Porsche.
User Manual for Mint—The Spice Document Preparation System.
 Spice Document, Carnegie Mellon University Computer Science Department,
 February, 1984.
- This document describes version 2B(12) of Mint, a document preparation system that has been written as part of the Spice project. Mint has been written as a research vehicle for exploring document preparation and interactive document preparation in particular. Although the current version of Mint does not have interactive features, it is nonetheless a usable tool which is suitable for release to a wider community for use and evaluation. In making this release, I am making a commitment to providing a stable and maintained system.
- The document is organized as follows. This introduction provides an overview of the system and gives operating information; it is followed by a brief review of Mint for Scribe users. The information provided should be sufficient to allow the casual user to prepare documents on the Perq of the same quality as those produced by Scribe. More detailed information can be found in the Reference Manual.
- [Ball 81] Ball, J.E.
 Canvas: The Graphics Package for the Spice Personal Timesharing System.
 In *Proceedings of Computer Graphics 81*, October, 1981.
- [Barbacci 81] Barbacci, M.R.
 Syntax and Semantics of CHDLs.
 In *Proceedings of the Fifth International Conference of Computer Hardware Description Languages and their Application*, North-Holland Publishing Company, September, 1981.
- [Barbacci 82a] Barbacci, M.R.
Intermediate Representation for the Spice Ada+ Compiler.
 Spice Document 138, Carnegie Mellon University Computer Science Department,
 September, 1982.
- This document describes the format of the Ada+ parse trees created by the Ada+ compiler although their lifetime is not restricted to that of a compilation: parse trees can be stored in separate files and can be manipulated by other programs. In particular, when Ada+ is parsing the context specifications of a package (currently the only compilation units allowed), it looks for both a text form (extension '.ada') and a parse tree form (extension '.gdb') of the imported package specifications. If a parse tree is not older than the text than Ada+ reads the parse tree directly, thus saving the entire lexical analysis phase.
- This document describes the format of files containing the Ada+ parse trees (we will refer to these as GDB files), and not the format of the parse trees as they exist during compilation. The file contains a subset of the information produced during a compilation since some of it is irrelevant or non-transportable across compilations.
- [Barbacci 82b] Barbacci, M.R.
The Ada+ Programming Environment.
 Spice Document 139, Carnegie Mellon University Computer Science Department,
 September, 1982.

- [Barbacci 82c] Barbacci, M. R. and D. P. Siewiorek.
The Design and Analysis of Instruction Set Processors.
McGraw-Hill Book Company, NY, 1982.
- [Barbacci 82d] Barbacci, M. and D.P. Siewiorek.
The Design and Analysis of Instruction Set Processors.
McGraw-Hill Book Company, 1982.
- [Barbacci 83a] Barbacci M.R., T.D. Newton, and R.G. Stockton.
The Ada+ Project.
Spice Report 162, Carnegie Mellon University Computer Science Department,
December, 1983.
Also appears in *Proceedings of the Third Ada UK Annual Conference: Ada Implementation and Early Experience*, January 1984.
This paper describes the motivation, status, and plans for the Ada+ Programming Environment being developed as part of the Spice project in the Department of Computer Sciences at Carnegie Mellon University. The goals of the project are the integration of programming tools through the use of a common intermediate representation and the use of multiple code generators and associated engines.
- [Barbacci 83b] Barbacci, M.R.
Hardware Description Languages,
In Ralston, *Encyclopedia of Computer Science*. Van Nostrand Reinhold Co., 1983.
- [Barbacci 84a] Barbacci, M.R., W.H. Maddox, T.D. Newton, and R.G. Stockton.
The Ada+ Data Base.
Spice Report 161, Carnegie Mellon University Computer Science Department,
September, 1984.
This document describes the intermediate form used by the Ada+ system. The intermediate form created by the Ada+ compiler is used during all phases of the compilation and is stored in a data base from which it can be retrieved for later use. Potential users are the compiler itself, (when processing context specifications), the linker, and the debugger (at run time). Its use is not limited to these subsystems. Version control and configuration management programs can also use the data base.
The structures used in the data base are identical to those used during compile time, except that (virtual memory) pointers between compile time data structures are translated into a compilation independent format. The format and the translation process are described later on.
- [Barbacci 84b] Barbacci, M.R., R.V. Baron, and M.R. Thompson (eds.).
Spice Commands and Utilities.
Spice Document , Carnegie Mellon University Computer Science Department,
August, 1984.
This document provides a description of how to use almost all of the Spice commands and utilities. You will find that some of the more powerful commands and utilities are fully documented in their own separate manuals and have only cursory documentation here.

- [Barbacci 84c] Barbacci, M.R. (ed.).
Introduction to the Spice Users' Manual.
 Technical Report, Carnegie Mellon University Computer Science Department,
 August, 1984.
- Spice stands for Scientific Personal Integrated Computing Environment. It is a project within the Computer Science Department of Carnegie-Mellon University with the goal of providing the software for a computing environment based on powerful personal computers. The environment that results will be able to replace the current time-shared systems for the majority of the computing needs of the department, and it is expected that it will serve these needs into the 1990's. The project is now making a public release of S5 version software. This software executes on Perq Systems Corporation PERC computers, which are being used as interim development machines.
- This release represents significant steps toward our goal. It contains major improvements relative to earlier versions. In particular, there are advancements in terms of performance and coherence, as well as new tools, such as Mint and Sapphire.
- [Clarke 82a] Clarke, E.M., A.P. Sistla, N. Francez and Y. Gurevich.
 Can Message Buffers be Characterized in Linear Temporal Logic?
 In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, August, 1982.
- [Clarke 82b] Clarke, E.M. and C.N. Nikolaou.
 Distributed Reconfiguration Strategies for Fault Tolerant Multiprocessor Systems.
IEEE Transactions on Computers(C-31), August, 1982.
- [Clarke 83] Clarke, E.M., S. German, and J.Y. Halpern.
 On Effective Axiomatizations of Hoare Logics.
Journal of the ACM, July, 1983.
- [Colwell 83] Colwell, R.P., C.Y. Hitchcock, and E.D. Jensen.
 A Perspective on the Processor Complexity Controversy.
 In *Proceedings, International Conference on Computer Design*, October, 1983.
- [Dannenberg 82] Dannenberg, R.B.
Resource Sharing in a Network of Personal Computers.
 Technical Report CMU-CS-82-152, Carnegie Mellon University Computer Science Department,
 December, 1982.
- As networks of personal computers are developed to replace centralized time-shared systems, the need for sharing resources will remain, but the solutions developed for time-sharing will no longer be adequate. In particular, the sharing of network resources is complicated by issues of security and autonomy, since a network of personal computers may be composed of nodes that are completely controlled by their owners.
- To facilitate sharing in this sort of environment, an operating system component called the *Butler* is proposed. As a *host*, the Butler is responsible for administering a sharing policy on its local machine. This includes authenticating sharers, granting rights in accordance with a locally established policy, and creating execution environments for *guests*.
- A number of applications for Butler are described: these fall into the categories of

information exchange, load distribution, and computational parallelism. A prototype Butler has been constructed and used in real application demonstrating computational parallelism, and the prototype has also demonstrated the deportation of processes.

[Fahlman 84] Fahlman, S.E. and S.P. Harbison.

The Spice Project,

In Barstow, Sandewall, and Shrobe, *Interactive Programming Environments*. McGraw-Hill, 1984.

Timesharing evolved as a way to provide users with the power of a large interactive computer system at a time when such systems were much too expensive to dedicate to a single individual. Recent advances in hardware technology are opening up new possibilities. The level of capital investment which today provides each user with a small slice of a time-shared machine and a crude CRT terminal will, by the mid-1980's, provide the same user with his or her own powerful machine, far more powerful than today's microprocessors and equipped with such features as high-resolution color graphics and audio I/O devices. This development will enable the user to avoid many of the compromises and limitations inherent in timesharing. New high-speed network technologies make it possible to move to this personal computing environment without foregoing the attractive features of timesharing: shared information, good inter-user communication, and the sharing of expensive peripherals.

The Spice project at Carnegie-Mellon University is one attempt to explore the possibilities of this new world. The goal of Spice is to develop an integrated research computing environment based on a network of very powerful personal computers. Machines of this power may be too expensive for widespread individual use today, but we believe that by the mid-1980's they will be easily affordable by most researchers.

[Giuse 84] Giuse, D.

Update: A File Transfer Facility.

Spice Document S165, Carnegie Mellon University Computer Science Department, August, 1984.

Update is a program for transferring sets of files between the Perqs and a remote host. Update will run between a Perq and any UNIX VAX. However, Spice *primarily* uses the CFS and Spice Vaxen. Update allows Spice developers to store on a VAX software that has been created on a Perq, and it allows users to transfer the newest version of Spice software from a VAX to their Perqs. Update's special feature is that it allows users to transfer only those files that are different from the ones already on their Perqs. This is accomplished by comparing the date of creation of each local file (on the Perq) in a set of files (SOF) with the creation date of remote files (those on a VAX). Because Update minimizes ethernet traffic and facilitates standardization of Spice software versions, it is used instead of Cmuftp for transfer of Spice software.

[Habermann 84] Habermann, A.N. and D.E. Perry.

Ada for Experienced Programmers.

Addison-Wesley Publishing Company, 1984.

The goal of this book is the presentation of the major features of the Ada programming language and their relevance to software engineering. Since concepts such as data abstraction, exception handling and concurrency are of fundamental importance to

the design and maintenance of software systems, we will explain in detail how Ada's facilities support such concepts. We do this by discussing a series of non-trivial example programs that exhibit the typical use of the relevant language constructs. The examples are chosen on the basis of their suitability in illustrating specific language features. Software engineering issues are taken into account, but are not the primary motivation for selecting the particular examples. Our goal is achieved if the examples demonstrate to what extent the Ada language supports good programming style in software engineering.

[Harbison 82a] Harbison, S.P. .

An Architectural Alternative to Optimizing Compilers.

In *Proceedings of the Symposium on Architectural Support for Programming Languages and Systems*, Pages 57-65. March, 1982.

Programming languages are designed to make *programming* productive. Computer architectures are designed to make *program execution* efficient. Although architectures should be designed with programming languages in mind, it may be as inappropriate to make the computer execute the programming language directly as it is to make the programmer use machine language. It is the compiler's job to match the programming language and the computer architecture, and therefore making compilers efficient and easy to write are important design goals of a complete hardware/software system. This paper summarizes research completed in 1980 on a computer architecture, *TM*, that takes over some of the more burdensome tasks of optimizing compilers for high-level-languages (HLL's), performing these tasks dynamically during the execution of the object program. This is a different approach to making compilers efficient than is commonly taken; more common approaches include devising more efficient optimization algorithms, being clever about when to do optimizations, and building the compilers semi-automatically.

[Harbison 82b] Harbison, S.P.

The Structure and Communication of Spice Objects.

Technical Report , Carnegie Mellon University Computer Science Department, April, 1982.

[Hayes 84] Hayes, P., R. Lerner, and P. Szekely.

Cousin Users Manual.

Technical Report Spice Project, Carnegie Mellon University Computer Science Department, August, 1984.

COUSIN is a program that provides uniform, cooperative, graphical, command interfaces for a variety of Spice applications. This manual describes the COUSIN interface system from the viewpoint of the end user of applications which use COUSIN to provide their user interface. If you wish to construct a COUSIN interface to one of your own applications, you will also need the COUSIN documentation for application builders which can be found in the Spice Programmers Manual.

COUSIN interfaces employ a form-based model of communication. Each application has an associated form analogous to the kind of business form in which information is entered by filling in blanks, or circling alternatives. The fields of the form correspond to the various pieces of information that the user and application need to exchange during an interactive session including input parameters, output from the application, and commands to the application. Forms of this kind show the user what his options are and provide a simple yet powerful interface through

which COUSIN can provide error detection and correction services.

- [Haynes 82] Haynes, L.S., R.L. Lau, D.P. Siewiorek and D.W. Mizell.
A Survey of Highly Parallel Computing.
Computer 15, January, 1982.
- [Hibbard 81] Hibbard, P., A. Hisgen, J. Rosenberg, M. Shaw and M. Sherman.
Studies in Ada Style.
Springer-Verlag Publishers, 1981.
- [Hibbard 82a] Hibbard, P.G. and B. Leverett.
An Adaptive System for Dynamic Storage Allocation.
Software Practice & Experience(12), 1982.
- [Hibbard 82b] Hibbard, P.G. and T.L. Rodeheffer.
Optimizing for a Multiprocessor: Balancing Synchronization Costs Against Parallelism in
Straight-Line Code,
In M. Dezani-Ciancaglini and U. Montanari, *Lecture Notes in Computer Science*, Pages
194-211. Springer-Verlag Publishers, 1982.
- [Hibbard 82c] Hibbard, P.G.
Spice: An Exercise in Personal Scientific Computing.
Spice Report , Carnegie Mellon University Computer Science Department,
1982.
- [Hibbard 82d] Hibbard, P.G., R. Whiteside, and N.S. Ostlund.
Systolic Algorithms for Monte Carlo Simulations.
In *Proceedings of the Third International Conference on Distributed Computing Systems*,
1982.
- [Hibbard 84] Hibbard, P.
Mint Reference Manual.
Spice Document , Carnegie Mellon University Computer Science Department,
August, 1984.
This document describes version 2B(12) of Mint, the Spice Document Formatter. This
is an early draft, and not all the facilities of Mint are accurately described. The
whole of the document has been produced by Mint and DP, executing on a Perq.
- [Horowitz 84] Horowitz, M., D. Nichols, E. Smith.
Mercury.
Spice Document , Carnegie Mellon University Computer Science Department,
August, 1984.
Mercury is the electronic mail system for Spice. Mercury provides commands for read-
ing mail, composing new mail, and organizing old mail. Mercury under Spice is
almost identical to Mercury under UNIX and is very similar to the RdMail system
on Tops-10 systems. The Spice system provides a transport mechanism known as
Mailman for getting electronic mail to and from a Perq. The Mailman program is
described in the *Spice Commands and Utilities Manual*. Mailman is run as a back-
ground server and, except for initialization, will not bother the user.
- [Jensen 83] Jensen, E.D.
Decentralized Resource Management for Embedded Computers.
In *Proceedings of the AIAA Conference on Computers in Aerospace IV*, October, 1983.

[MacLachlan 84] MacLachlan, R.

Hemlock User's Manual.

Technical Report, Carnegie Mellon University Computer Science Department,
August, 1984.

This document describes the Hemlock text editor, as of version 0.99(24). Hemlock is a customizable, extensible text editor whose initial command set closely resembles that of ITS/TOPS-20 EMACS. Hemlock is written in the Spice Lisp implementation of Common Lisp, and can be ported to other implementations.

[Newton 84]

Newton, T.D., M.R. Barbacci, W.H. Maddox, R.G. Stockton.

User's Guide to the Ada+ Compiler.

Spice Document 173, Carnegie Mellon University Computer Science Department,
October, 1984.

This document describes the first public version of the Ada+ compiler, which runs on the VAX and on the Perq (under Accent), producing code for the Perq. We are releasing the compiler at this time so that it can be used by students taking the IC course on Ada and any other willing guinea pigs.

The compiler implements full Ada, except for tasking and fixed-point types. Most of the basic features work. Currently it compiles programs at about half the speed of the Perq Pascal compiler, produces intermediate files that are about ten times the size of the source files, and produces code that is about as good as the code produced by Perq Pascal.

[Ostland 82]

Ostlund, N.S., P.G. Hibbard, and R.A. Whiteside.

A Case Study in the Application of a Tightly-Coupled Multiprocessor to Scientific Computations,

In B. Alder, S. Fernbach and, M. Rotenberg, *Parallel Computations*. Academic Press, 1982.

[Piloty 83]

Piloty, R., M.R. Barbacci, D. Borriore, D. Dietmeyer, F. Hill, and P. Skelly.

CONLAN Report.

Lecture Notes in Computer Science 151, 1983.

[Rashid 84]

Rashid, R.F.

Accent: A Distributed Operating System for a Network of Scientific Personal Computers.

In *Proceedings of the Convention Informatique 84*, September, 1984.

Accent is a message based distributed operating system kernel developed at Carnegie-Mellon University to support a large network of personal scientific workstations. Accent combines a network transparent interprocess communication facility with sophisticated virtual memory management to allow copy-on-write transfer of data between processes on the same processor and copy-on-reference data transfer between processes on different computers on a local area network. Accent differs from other network operating system efforts such as Locus Computer Corporation's LOCUS and Apollo Corporation Aegis in that all system and user provided services (even kernel functions) can be distributed transparently on the network. Despite its generality, the performance of Accent is comparable to a more traditionally constructed operating system on the same processor.

Accent is currently being used on a network of 150 PERQ Systems Corporation PERQ and PERQ2 computers interconnected via 10 MHz and 3MHz Ethernet. The Accent interprocess communication facility is available on the Computer Science Department's 40 VAX computers as a modification to Berkeley UNIX (4.1bsd and 4.2 bsd). Network operating system services on both VAX and PERQ computers

include network interprocess communication, remote process invocation, transparent file access, printing services, network graphics and network name services. Accent is being marketed for PERQ and PERQ2 computers in the United States by PERQ Systems Corporation under license from Carnegie-Mellon University. Work is underway to port Accent to other workstations as well as a large multiprocessor.

[Sansom 84a]

Sansom, R.

SPICE System Programmers Guide.

Spice Document, Carnegie Mellon University Computer Science Department, August, 1984.

This document provides an overview of how to write programs which use the ACCENT operating system. Thus it explains how to use the server processes running under Spice, how to create your own processes and how to talk between your own processes. It does not attempt to explain how to make modifications to the operating system except for giving the procedure for constructing a new system image.

[Sansom 84b]

Sansom, R.

AccUnix - Accent style IPC under UNIX.

Spice Document, Carnegie Mellon University Computer Science Department, August, 1984.

An explanation of how to use the facilities available under UNIX for doing Accent style IPC. In addition, a description of how to use the UNIX server for communication with other machines.

[Satyanarayanan 81]

Satyanarayanan, M.

A Study of File Sizes and Lifetimes.

Technical Report CMU-CS-81-114, Carnegie Mellon University Computer Science Department, April, 1981.

An investigation of the size and lifetime properties of files on the primary computing facility in the Department of Computer Science at Carnegie-Mellon University is presented in this paper. Three key issues are examined: the effect of migration on file characteristics, the effect of file type on file characteristics, and the correlation between file sizes and lifetimes. Analytical models that fit the observed data are derived using two alternative techniques.

[Satyanarayanan 83]

Satyanarayanan, M.

A Methodology for Modeling Storage Systems and Its Application to a Network File System.

Technical Report CMU-CS-83-109, Carnegie Mellon University Computer Science Department, March, 1983.

The central question addressed by this dissertation is: Can one structure the modeling of complex memory systems in such a way as to permit the use of standard, off-the-shelf software for modeling the behavior of individual memory devices?

The highlights of the research reported in the dissertation are:

- The development of a methodology that permits simulation models for memory systems to be built from performance models of the com-

ponents, and the description of a tool based on this methodology.

- A demonstration of the side scope of applicability of this methodology by examples drawn from the primary, secondary and tertiary levels of the memory hierarchy.
- An implementation of the tool, and measurements to estimate the overheads imposed by it.
- The use of the tool in the analysis of an actual memory system.
- The dissertation thus provides, in a constructive manner, an affirmative answer to the question posed at the beginning of this document.

[Schwans 84] Schwans, Karsten.

Tailoring Software for Multiprocessor Systems.

PhD thesis, Carnegie Mellon University Computer Science Department, 1984.

Multiple processor systems are becoming increasingly common. However, their use remains difficult due to a lack of knowledge concerning the development of parallel application programs. In addition, contrary to popular predictions of 'cheap and plenty' resources, efficient management of distributed processor and memory resources remains of critical importance to the successful use of these systems.

The contributions of this thesis are twofold. First, we design and implement a programming environment for multiple processor applications, called the TASK system. Second, we discuss the integration of policies and mechanisms for resource management into the TASK system.

In TASK, application programs are written in terms of abstractions offered by the operating system used for program execution. As a result, once an application program is written, its execution requires few additional efforts by the application's programmer. Programs are written in two languages. The TASK language, designed and implemented as part of this thesis, is used to describe the logical structure of an applications program, and an existing, algorithmic language is employed to implement the applications algorithms. The construction of an executable version of an application of the TASK and algorithmic language is automated. Such construction includes automatic linking and loading as well as the allocation of resources to the individual components of the application program.

Programmers guide the allocation of hardware resources to program components by stating high-level directives in TASK programs. To identify suitable directives and to develop procedures that automatically perform resource allocation based on these directives, we develop a model of multiple processor software and hardware, called the proximity model. The model, the directives, and the resource allocation procedures are tested by experimentation with application programs on the Cm* multiprocessor.

[Sha 83]

Sha, D., E.D. Jensen, R.F. Rashid, and J.D. Northcutt.

Distributed Cooperating Processes and Transactions,

Synchronization, Control, and Communication in Distributed Computing Systems., 1983.

[Snow 81]

Snow, E.A. and D.P. Siewiorek.

Implementation and Performance Evaluation of Computer Families.

*IEEE Transactions on Computers*C-30(6), 1981.

- [Spector 82a] Spector, A.Z.
Performing Remote Operations Efficiently on a Local Computer Network.
Communications of the ACM 25(4), April, 1982.

A communication model is described that can serve as a basis for a highly efficient communication subsystem for local networks. The model contains a taxonomy of communication instructions that can be implemented efficiently and can be a good basis for interprocessor communication. These communication instructions, called remote references, cause an operation to be performed by a remote process and, optionally, cause a value to be returned. This paper also presents implementation considerations for a communication system based upon the model and describes an experimental communication subsystem that provides one class of remote references. These remote references take about 150 microseconds or 50 average instruction times to perform on Xerox Alto computers connected by a 2.94 megabit Ethernet.

- [Spector 82b] Spector A.Z. and P. Schwarz.
Synchronizing Shared Abstract Types.
Technical Report CMU-CS-82-128, Carnegie Mellon University Computer Science Department,
July, 1982.

This paper discusses the synchronization issues that arise when transaction facilities are extended for use with shared abstract data types. A formalism for specifying the concurrency properties of such types is developed. This formalism uses dependency relations that are defined in terms of an abstract type's operations. It requires that the definition of an abstract type state whether or not cycles involving these relations should be allowed to form. Directories and two types of queues are specified using the technique, and the degree to which concurrency is restricted by type-specific consistency properties is exemplified. A locking mechanism is described that permits implementations to make use of this type-specific information to approach the limits of concurrency.

- [Spector 83] Spector, A.Z. and P. Schwarz.
Transactions: A Construct for Reliable Distributed Computing.
Operating Systems Review 17(4), April, 1983.

Transactions have proven to be a useful tool for constructing reliable database systems and are likely to be useful in many types of distributed systems. To exploit transactions in a general purpose distributed system, each node can execute a transaction kernel that provides services necessary to support transactions at higher system levels. The transaction model that the kernel supports must permit arbitrary operations on the wide collection of data types used by programmers. New techniques must be developed for specifying the synchronization and recovery properties of abstract types that are used in transactions. Existing mechanisms for synchronization, recovery, deadlock management, and communication are often inadequate to implement these types efficiently, and they must be adapted or replaced.

- [Steele 84] Steele, G.
Common Lisp: The Language.
Digital Press, 1984.

- [Thompson 85] Mary R. Thompson, Robert D. Sansom, Michael B. Jones Richard F. Rashid.
Sesame: The Spice File System.
 Technical Report CMU-CS-85-172, Carnegie Mellon University Computer Science Department,
 November, 1985.
 Sesame provides several distinct but interrelated services needed to allow protected sharing of data and services in an environment of personal and central coputers connected by a network. It provides a smooth memory hierarchy between the local secondary storage and central file system. It provides a global name space and a global user authentication protocol.
- [Uehara 83] Uehara, T. and M.R. Barbacci (eds.).
Proceedings of the Sixth International Symposium on Computer Hardware Description Languages and their Applications.
 North Holland Publishing Company, 1983.
- [Whiteside 82] Whiteside, R.A., P.G. Hibbard, and N.S. Ostlund.
 A Systolic Algorithm for Monte Carlo Simulations.
 In *Proceedings of the Third International Conference on Distributed Computer Systems*, 1982.
- [Wright 84] Wright, K.
Oil: The Spice ASCII Editor
 1984.
 Internal Working Document of the Carnegie Mellon Computer Science Department.

3. Research in Image Understanding

Image Understanding (IU) research aims at understanding and constructing AI systems that can perceive their external world through visual images. IU systems typically operate in conjunction with other, larger systems that use perceptual input. Thus, we evaluate IU systems ultimately by how much they contribute toward visual capabilities the larger systems need to accomplish particular tasks.

The primary objective of CMU's IU research effort over the past few years has been to develop techniques and systems that will lead to a successful demonstration of image understanding concepts over a wide variety of tasks. Our work spans three interrelated areas:

- *Understanding three-dimensional shapes:* developing theories and techniques that distinguish three-dimensional shapes from two-dimensional images and permit a system to comprehend the structure of the visual environment.
- *Modeling a three-dimensional world:* applying specialized algorithms and control structures to model three-dimensional urban scenes from aerial photographs and range images.
- *Applications of three-dimensional methods:* inventing custom architectures and programming structures that can realize vision techniques and structures efficiently.

In addition to our work in these areas, we have contributed several expert system and programming packages to the SRI Testbed facilities.

3.1. Understanding Three-Dimensional Images

One fundamental challenge in vision research is getting machines to discern three-dimensional shapes represented as two-dimensional images. At CMU we have made progress in solving several aspects of this problem. Our contributions include defining fundamental gradient space properties, applying shadow-derived information, exploring generalized cylinder applications, and developing a new analysis technique for stereo image pairs.

3.1.1. Gradient space theory

Kanade and Shafer [Shafer 82a] have developed and summarized important properties of Mackworth's gradient space. Their introduction and use of vector (edge) gradients as well as surface gradients provide concise notation for several results. They explored properties including orthographic and perspective projections; gradient definitions; gradient space implications of vectors belonging to one or more surfaces and of several vectors contained on a single surface; and relationships among vanishing points, vanishing lines, and the gradient space. The vision research community has used their work as a study guide and reference.

3.1.2. Shadow geometry theory

The shadow geometry theory applies a gradient space approach to three-dimensional shape inference. It exploits image shadows that, in previous methods, had inhibited finding surface orientations. Shafer [Shafer 82b] has applied the theory to find surface orientations in polyhedra and generalized cylinders.

The shadow geometry theory poses a "basic shadow problem" where a single light source illuminates a surface and casts its shadow on a second, background surface. Six parameters specify the problem: the surface orientations (two parameters each) and the source vector direction. Given a line drawing that identifies shadow regions, the shadow geometry strategy uses shadow shape to generate constraints on possible surface orientations. The strategy identifies "illumination surfaces" bounded by illumination vectors, assigns Huffman-Clowes line labels to surface edges, and applies the corresponding constraints in gradient space. Given any three parameters, the method can determine the remaining unknowns.

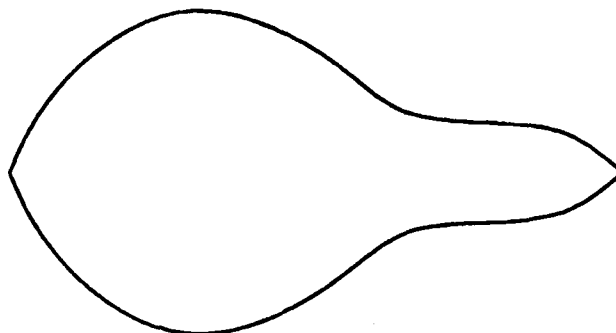
Our work has also extended the basic problem to polyhedral and generalized cylinder analyses and yielded benefits in both cases over previously used shape inference techniques. Shadow geometry mitigates one of the classic problems in inferring shape from real images: low edge contrast within shadows. When analyzing a polyhedron image via shadow geometry, detecting all shadow edges becomes unnecessary. For a generalized cylinder's curved surface, shadow geometry reveals surface orientation along a strip that cuts across the surface image and provides information that complements curved-surface interpolation techniques.

3.1.3. Generalized cylinders

Binford's generalized cylinders form a commonly-used shape representation scheme in computer vision, but research using the cylinders has been entirely based on heuristics, due to a lack of mathematical understanding and precise definition. We have investigated the properties of generalized cylinders and presented a comprehensive framework including several subclasses. Our new definition, based on a mathematical model, is more general than past definitions.

One difficulty with using generalized cylinders lies in calculating three-dimensional descriptions from images of their occluding contours (outlines). Shafer and Kanade [Shafer 83a] studied this problem and obtained their strongest results for solids of revolution, a generalized cylinder subclass. They produced a closed-form method for analyzing solids-of-revolution image contours and demonstrated that a picture of the contours is ambiguous (see Figures 3-1 and 3-2), with one degree of freedom related to the angle between the line of sight and the solid's axis. Further analysis, such as applying the shadow geometry method, can resolve this ambiguity and provide a unique 3-D interpretation of a solid-of-revolution image.

(a) solid of revolution, side view



(b) image of solid seen at 45 degrees

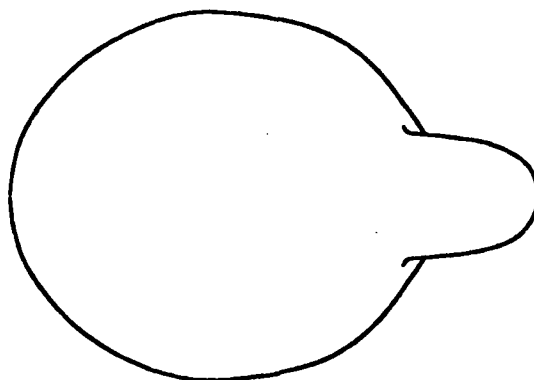


Figure 3-1: Solid-of-revolution contours

3.1.4. Image matching by two-stage dynamic programming

The key process in obtaining depth information via stereopsis is a search that identifies corresponding points in left and right images. Then, given a camera model specifying the camera positions, triangulation will yield the depth. Though the general correspondence problem involves searching an entire image, we can further exploit the camera model and reduce the process to a set of scanline-scanline correspondence sub-problems. That is, after rectifying an image pair so that the epipolar lines become horizontal scanlines, we can restrict the search to a single pair of corresponding scanlines.

In edge-based stereo techniques, the search concentrates on points that delineate edges. The *intra-scanline* method seeks edge-point pairs. The problem in this situation becomes one of finding a match-path on a two-dimensional search plane whose axes are the right and left scanlines. Well-known dynamic programming techniques handle the 2-D search efficiently. Where an edge extends across scanlines, vertically connected image points provide additional consistency constraints over the 2-D search plane. Our research exploits mutual correlations between scanlines by adding a second, *inter-scanline* search to find consistencies among

(c) Possible interpretations of image at various viewing angles.

Thick lines are visible portions; thin lines are estimated
interpretations are scaled to same horizontal size.

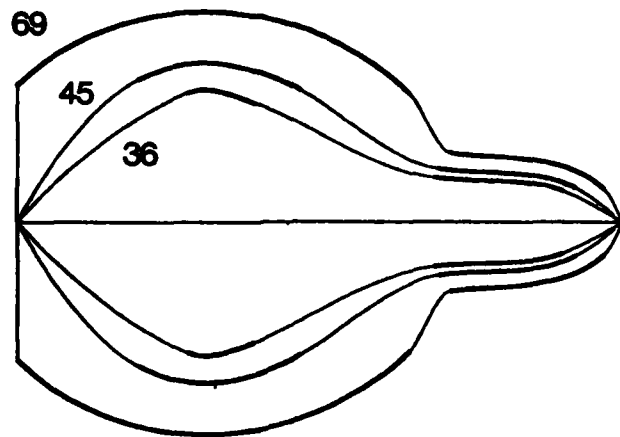


Figure 3-2: Interpreting a solid-of-revolution image

scanlines. We have produced an efficient stereo image-matching algorithm that simultaneously searches both within and between image scanlines [Ohta 83].

The algorithm uses edge-delimited intervals as the elements to match. Pursuing both intra- and inter-scanline searches, we cast the problem as a three-dimensional search for a surface that best matches intra-scanline edge points while conforming to inter-scanline constraints. Our method reduces computation to a feasible amount and we have successfully processed urban aerial photographs containing tall buildings, roads, and trees.

3.2. Modeling a Three-dimensional World

A vision system must be capable of more than classifying and segmenting images or identifying objects in images to perform navigation, change detection, model-based interpretation, and other tasks. It must be able to generate a three-dimensional description of the scene, or model a 3-D world. In the last few years we have developed automatic techniques for building 3-D descriptions of scenes and objects from aerial images and range data.

3.2.1. Modeling from aerial photography

It is difficult to build a complete description of a complex scene from a single view, because one view provides only partial information about the scene. Many surfaces may be occluded and some visible portions may be difficult to recover because of shadows, highlights, and oblique viewing angles. The 3-D Mosaic System, developed and tested by Herman and Kanade [Herman 83a], is a model-based vision system that incrementally acquires a 3-D description of urban scenes from aerial photographs. Our system uses multiple images obtained from multiple viewpoints under different conditions and builds a coherent model by combining partial 3-D information from each view. This approach aids interpretation in two ways. First, surfaces occluded in one image may become visible in another. Second, surface features that are difficult to analyze and interpret in one image may become more apparent in another because of different viewpoint and/or lighting conditions. The 3-D description serves as a model during its construction: at each step, the current description determines which part of a scene, at which angle, should be taken from the next image.

3.2.2. Modeling from 3-D Range Imagery

Extracting 3-D information from 2-D image data poses a formidable challenge, partly because the system must infer distance from intensity. Recent technological advances, however, permit *direct* extraction of the third dimension in the form of range images. In addition to difficulties shared with 2-D image processing, range data pose their own special problems: extracting useful features, handling noise, etc.

Smith and Kanade [Smith 84] developed a system that produces object-centered 3-D descriptions, beginning with 3-D data obtained by a light-stripe rangefinder. The thrust of this work is data-driven, bottom-up, autonomous processing that generates object descriptions from complicated scenes without referring to specific pre-stored object models.

Beginning with iconic range data, our system generates descriptions while moving up a hierarchy of contour, surface, and object to scene. In so doing, it exploits coherent relationships such as symmetry, collinearity, and coaxiality among lower-level elements, to hypothesize upper-level elements. This strategy is justified because those coherent relationships do not usually occur accidentally [Kanade 81].

We focused our efforts on the use of occluding contours (outlines), which we can extract reliably from the light-stripe range data. First, the program extracts, segments, and classifies contours. From the coherencies among them, such as parallelism, surfaces are hypothesized and represented as conic surfaces (pipes, cones, and planes). The program then confirms or refutes surface hypotheses by their ability to account for observed surface area. It examines coherencies among the verified surfaces, such as axis intersections, to form hypotheses of surface groups. Finally, the program compares surface groupings from multiple scenes; if a

similar structure repeatedly occurs, it identifies the structure as an object. The program has been tested on several scenes that include pans, cups, shovels, and polyhedra.

Tomita and Kanade developed a 3-D shape matching program that matches object models with the scene descriptions obtained from the range image by finding appropriate coordinate transformations. It hypothesizes a transformation by initially matching a few scene features with model features, then tests the transformation with the remaining features for verification. The object models can be generated interactively from the example scenes. In our program, object models can represent not only objects with rigid, fixed shapes, but also objects with inter-part articulations such as rotational joints or linear-slide joints. This representation also allows the program to process objects with three-dimensional symmetry (e.g., a cup with a handle) unambiguously.

The IUS Group has been developing other facilities that acquire, process, and display 3-D range images to aid research on 3-D range data analysis. The facilities include a data acquisition system for an industrial setting and a library of programs for various tasks. The library includes software for boundary detection, 3-D curve segmentation, 3-D edge detection, and a 3-D display program with rapid data/description overlay graphics.

3.3. Applications in Digital Mapping and Photointerpretation

The development of intelligent spatial databases addresses two problems in digital mapping. First, from a database perspective, the explosive increase in the availability of imagery and image-related information makes finding a small piece of relevant data increasingly difficult. Storing tens of thousands of images on-line is useless unless a user can quickly locate one interesting feature or landmark in many different images simultaneously. Second, symbolic indexing and addressing of images for automated analysis requires many of the same techniques as interactive analysis, except that in the latter, a human provides the guidance. Facilities such as on-line image/map databases, signal and symbolic indexing of natural and man-made features, and spatial reasoning can be viewed in the short term as semi-automated tools that increase human photointerpreters' productivity. In the long term, the facilities provide a knowledge base for automated systems that can perform detailed analysis, including change detection and automatic map description updating.

3.3.1. An integrated spatial database

The IUS Group has developed the Map Assisted Photointerpretation System (MAPS), a large scale image/map database system [McKeown 83]. The first such system to work with a complex urban environment, MAPS contains approximately 100 high resolution aerial mapping images, a digital terrain (elevation)

database, and map databases provided by the USGS and DMA. The system integrates these databases using large-scale, high-resolution imagery to cover approximately a 150 square mile area centered over the District of Columbia. Users can interact with a high resolution image display and query the database for names, descriptions, and spatial relationships among natural and man-made map entities. Our research concerns evaluating a hierarchical spatial representation to constrain the search in large databases, applying spatial knowledge to navigate within a map database, and supporting complex queries that arise in cartography. Dynamic symbolic and signal access to the image/map database, detailed semantic descriptions of man-made and natural features, generalized geometric computations of map feature relationships, and an intelligent window-based image display manager distinguish MAPS from other work in this area.

Spatial databases require many query capabilities and access mechanisms not found in more traditional database applications. The following is a brief list of some of the current capabilities in MAPS:

- *Geodetic Location* Query of map entities extracted from imagery in map coordinates rather than image coordinates.
- *Time* Selective ordering of imagery and map data based on acquisition date and time.
- *Map Feature Descriptions* Query of map entities based on full or partial descriptions of attributes such as shape, area, population, and user-defined classifications.
- *Access Methods* Multiple access methods are provided.

Template Matching—Match on factual templates.

Signal Access—Match on location.

Symbolic Access—Match by name.

Geometric Access—Match by spatial constraints.

Memo Functions—Look up facts before computing from database.

- *Spatial Decomposition* Limitation of query area using inherent spatial decompositions, such as political boundaries and neighborhoods.

3.4. Vision for Navigation

A mobile robot needs to know its motion and position relative to fixed objects, and must be able to track moving objects. We have explored three vision methods that use multiple images to track both the robot's motion and that of nearby objects. One method uses sequences of stereo image pairs, another exploits monocular image sequences, and a third uses an adaptation of optical flow techniques.

3.4.1. A visual navigation system for the CMU rover

The FIDO [Thorpe 84] visual navigation system guides a robot built by the CMU Mobile Robot Lab to a pre-defined location in a static environment. Our work extended and improved upon Moravec's work with stereo image pair sequences. We first re-implemented the Stanford Cart algorithms and then successively examined several of the most important components:

- A prototype *multiprocessor implementation* running on multiple VAXes showed that FIDO could be efficiently decomposed into several cooperating processes. With a high bandwidth communications channel or shared main memory, the multiprocessor version could run in as little as three seconds per step.
- Our examination of *interest operators*, the means for picking points to match from one image to the next, showed that changing low-level image-processing algorithms made much less difference than adding geometric constraints.
- We built a better *geometry module* that uses more of the available geometric constraint on the image matching process. This allowed us to drop from nine images at each step to two, decreasing run time to 30 CPU seconds per step while maintaining accuracy.
- We developed a new path planning algorithm, *Path Relaxation*, that explicitly takes into consideration the robot's field of view and the potentially conflicting goals of finding a shorter path and staying further away from objects.

Various versions of Fido were tested and demonstrated throughout 1983.

3.4.2. Obtaining camera motions from image sequences

Lucas and Kanade [Lucas 81] applied their "method of differences" technique to the problem of obtaining camera motion from an image sequence. The technique uses inter-image intensity differences and local intensity gradients to iteratively estimate camera motion. First iterations use images that have been smoothed with a low-pass filter, to get rough estimates for the parameters; later iterations use successively less-smoothed images to achieve increasingly higher resolution.

By applying simple image smoothing and pixel differences, instead of correlations, and using multiple resolution levels, our method avoids the expensive searches used by many other matching techniques. If there are fewer unknown parameters (as in motion confined to a plane or for a pair of stereo cameras), our technique will tolerate more error in initial parameter estimates and still converge to the correct solution. Lucas's thesis [Lucas 84] includes an analysis of various real and synthetic image examples.

3.4.3. Obtaining object motion from image sequences

We have adapted Horn's and Schunck's optical flow algorithm to the problem of determining arbitrary motions of an object from two-dimensional image sequences. The adapted algorithm allows for both gradual changes in an object's appearance in the image sequence and flow discontinuities at the object's boundaries.

Our algorithm [Cornelius 83] uses a procedure that creates velocity fields for estimating an object's velocity and brightness changes (i.e., x-ray thickness changes) in an image plane. The procedure computes velocities from a series of images, using information about the spatial and temporal brightness gradients. We have applied the method to x-ray images of an expanding ellipsoid and of a beating heart, and it can also be used with reflectance images.

3.5. Contributions to the SRI Testbed Facilities

DARPA and DMA have jointly established an integrated demonstration system "testbed," with SRI as the integrating contractor. The testbed will be used for demonstrating and evaluating the applicability of IU research to cartography. The testbed has a user interface that simulates a cartographic workstation environment, and includes a computer with display terminal, an image display with track ball, and a digitizing tablet. The system will support all major steps in mapmaking with continuously evolving automation. DMA will do most image digitization off-line. The following contributions to the IUS testbed by CMU have been documented in the DARPA/DMA Image Understanding Testbed User's Manual [Hanson 84]:

- Delivered the Phoenix region-segmentation program [Law 82] that has a flexible user interface and interactive graphics.
- Delivered all of our basic VAX software facilities for running image understanding programs, including modified versions of the CMU image access and Grinnell graphics packages. This code became the core of SRI's testbed facilities.
- Implemented an improved version of the Moravec stereo reconstruction algorithm for the testbed. This C version incorporates several improvements that increase speed and flexibility.

3.6. Bibliography

- [Cornelius 83] Cornelius, N.H. and T. Kanade.
Adapting Optical Flow to Measure Object Motion in Reflectance and X-Ray Image Sequences.
In Lee S. Baumann, Editor, *14th Proceedings of the DARPA Image Understanding Workshop*, Pages 257-266. DARPA Information Processing Techniques in conjunction with Computer Vision and Pattern Recognition Conference of the IEEE Computer Society, June, 1983.
Available as reprint no. AD-POO1 215 from Science Applications, Inc., MacLean, VA. Also available as CMU-CSD Technical Report CMU-CS-83-119.
This paper adapts Horn and Schunck's work on optical flow [3] to the problem of determining arbitrary motions of objects from 2-dimensional image sequences. The method allows for gradual changes in the way an object appears in the image sequence, and allows for flow discontinuities at object boundaries. We find velocity fields that give estimates of the velocities of objects in the image plane. These velocities are computed from a series of images using information about the spatial and temporal brightness gradients. A constraint on the smoothness of motion within an object's boundaries is used. The method can be applied to interpretation of both reflectance and x-ray images. Results are shown for models of ellipsoids undergoing expansion, as well as for an x-ray image sequence of a beating heart.
- [Hanson 84] Hanson, A.
DARPA/DMA Image Understanding Testbed User's Manual.
SRI International(277), January, 1984.
Version 1.1, contract no. MDA903-79-C-0588.
This manual is intended to help users of the Image Understanding Testbed system understand the structure and major features of the environment. Separate chapters are devoted to getting started as a new Testbed user, the Unix, Franz Lisp, and emacs programming systems, Testbed applications programs, and Testbed utility systems. Appendices contain descriptions of demonstrations, picture and graphics utilities, the file system structure, and some elementary programming examples.
- [Herman 83a] Herman, M., T. Kanade, and S. Kuroe.
The 3D MOSAIC Scene Understanding System.
In Alan Bundy, Editor, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence, August 8-12, 1983*, Pages 1108-1112. IJCAI, Inc., August, 1983.
Contact William Kaufmann, Inc., for reprint information.
A scene understanding system derives and represents information about a given scene for the purpose of various given tasks. The 3D Mosaic system incrementally derives a three-dimensional description of a complex urban scene from multiple images. The description, which we call a *scene model*, is intended to be useful for tasks such as matching, display generation, planning paths through the scene, and making other decisions dealing with the scene environment. This paper briefly describes the system and some experiments in acquiring and using the scene model. Further details may be found in [6,7].

- [Herman 83b] Herman, M.
 Monocular Reconstruction of a Complex Urban Scene in the 3D Mosaic System.
 In Lee S. Baumann, Editor, *14th Proceedings of the DARPA Image Understanding Workshop, June 23, 1983*, Pages 318-326. DARPA Information Processing Techniques Office, in conjunction with Computer Vision and Pattern Recognition Conference of the IEEE Computer Society., June, 1983.
 Available as reprint no. AD-POO1 220 from Science Applications, Inc., MacLean, Va.

A system for obtaining a surface-based, three-dimensional description of a complex urban scene from a single image is described, and an example involving an aerial photograph is provided. The author's approach exploits task-specific knowledge involving block-shaped objects in an urban scene. First, linear connected structures in the image are generated; these are meant to represent building boundaries. Next, the two-dimensional structures are converted into three-dimensional wire frames. Finally, an approximate surface-based description of the scene is generated from the wire frames. The monocular analysis system is a component of the 3D Mosaic Scene Understanding System.

- [Herman 84a] Herman, M., T. Kanade, and S. Kuroe.
 Incremental Acquisition of a Three-Dimensional Scene Model from Images.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6(3):331-340, May, 1984.

Also published as CMU-CSD Technical Report CMU-CS-82-139.

We describe the current state of the 3D MOSAIC project, whose goal is to incrementally acquire a three-dimensional model of a complex urban scene from images. The notion of incremental acquisition arises from the observations that 1) single images contain only partial information about a scene, 2) complex images are difficult to fully interpret, and 3) different features of a given scene tend to be easier to extract in different images because of differences in viewpoint and lighting conditions. In our approach, multiple images of the scene are sequentially analyzed so as to incrementally construct the model. Each new image provides information which refines the model. We describe some experiments toward this end. Our method of extracting 3D shape information from the images is stereo analysis. Because we are dealing with urban scenes, a junction-based matching technique proves very useful. This technique produces rather sparse wire-frame descriptions of the scene. A reasoning system that relies on task-specific knowledge generates an approximate model of the scene from the stereo output. Gray scale information is also acquired for the faces in the model. Finally, we describe an experiment in combining two views of the scene to obtain a refined model.

- [Herman 84b] Herman, M. and T. Kanade.
The 3D MOSAIC Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images.
 Technical Report CMU-CS-84-102, Carnegie Mellon University, Computer Science Department, February, 1984.

The 3D Mosaic system is a vision system that incrementally reconstructs complex 3D scenes from multiple images. The system encompasses several levels of the vision process, starting with images and ending with symbolic scene descriptions. This paper describes the various components of the system, including stereo analysis,

monocular analysis, and constructing and modifying the scene model. In addition, the representation of the scene model is described. This model is intended for tasks such as matching, display generation, planning paths through the scene, and making other decisions about the scene environment. Examples showing how the system is used to interpret complex aerial photographs of urban scenes are presented.

Each view of the scene, which may be either a single image or a stereo pair, undergoes analysis which results in a 3D wire-frame description that represents portions of edges and vertices of objects. The model is a surface-based description constructed from the wire frames. With each successive view, the model is incrementally updated and gradually becomes more accurate and complete. Task-specific knowledge, involving block-shaped objects in an urban scene, is used to extract the wire frames and construct and update the model.

[Kanade 81]

Kanade, T.

Recovery of the Three-dimensional Shape of an Object from a Single View.

Artificial Intelligence 17(1-3):409-460, August, 1981.

Given a single picture which is a projection of a three-dimensional scene onto the two-dimensional picture plane, we usually have definite ideas about the three-dimensional shapes of objects. To do this we need to use assumptions about the world and the image formation process, since there exist a large number of shapes which can produce the same picture.

The purpose of this paper is to identify some of these assumptions--mostly geometrical ones--by demonstrating how the theory and techniques which exploit such assumptions can provide a systematic shape-recovery method. The method consists of two parts. The first is the application of the Origami theory which models the world as a collection of plane surfaces and recovers the possible shapes qualitatively. The second is the technique of mapping image regularities into shape constraints for recovering the probable shapes quantitatively.

Actual shape recovery from a single view is demonstrated for the scenes of an object such as a box and a chair. Given a single image, the method recovers the three-dimensional shapes of an object in it, and generates images of the same object as we would see it from other directions.

[Law 82]

Law, K., S. Shafer, T. Kanade and D. Williams.

The Phoenix Image Segmentation System: Description and Evaluation.

SRI International (289), December, 1982.

Version 1.1, contract no. MDA903-79-C-0588, SRI Project # 1009.

Phoenix is a computer program for segmenting images into homogeneous closed regions. It uses histogram analysis, thresholding, and connected-components analysis to produce a partial segmentation, then resegments each region until various stopping criteria are satisfied. Its major contributions over other recursive segmenters are a sophisticated control interface, optional use of more than one histogram-dependent intensity threshold during tentative segmentation of each region, and spatial analysis of resulting subregions as a form of 'look-ahead' for choosing between promising spectral features at each step.

Phoenix was contributed to the DARPA Image Understanding Testbed at SRI by Carnegie-Mellon University. This report summarizes applications for which Phoenix is suited, the history and nature of the algorithm, details of the Testbed implementation, the manner in which Phoenix is invoked and controlled, the type

of results that can be expected, and suggestions for further development. Baseline parameter sets are given for producing reasonable segmentations of typical imagery.

- [Lucas 81] Lucas, B.D., and T. Kanade.
An Iterative Image Registration Technique With an Application to Stereo Vision.
In Lee S. Baumann, Editor, *Proceedings of the 12th DARPA Image Understanding Workshop*,
Pages 121-127. DARPA, April, 1981.
Also found in IJCAI '81 Proceedings, pp.674-679.

Image registration finds a variety of applications in computer vision. Unfortunately, traditional image registration techniques tend to be costly. We present a new image registration technique that makes use of the spatial intensity gradient of the images to find a good match using a type of Newton-Raphson iteration. Our technique is faster because it examines far fewer potential matches between the images than existing techniques. Furthermore, this registration technique can be generalized.

- [Lucas 84] Bruce D. Lucas.
Generalized Image Matching by the Method of Differences.
PhD thesis, Carnegie Mellon University, Computer Science Department, July, 1984.

- [McKeown 81] McKeown, D.M., Jr., and T. Kanade.
Database Support for Automated Photo Interpretation.
In Lee S. Baumann, Editor, *Proceedings of the DARPA Image Understanding Workshop*,
Pages 7-13. DARPA, April, 1981.

This paper is concerned with the use of a database to support automated photo interpretation. The function of the database is to provide an environment in which to perform photo interpretation utilizing software tools, and represent domain knowledge about the scenes being interpreted. Within the framework of the database, image interpretation systems use knowledge stored as map, terrain, or scene descriptions to provide structural or spatial constraints to guide human and machine processing. We describe one such system under development, MAPS (Map Assisted Photo interpretation System), and give some general rationales for its design and implementation.

- [McKeown 82a] McKeown, D.M., Jr.
Concept Maps.
In Lee S. Baumann, Editor, *13th Proceedings of the DARPA Image Understanding Workshop*,
Pages 145-153. Science Applications, Inc., MacLean, VA, September, 1982.
Also appears in the CMU-CSD technical report CMU-CS-83-117.

This paper describes a representational mechanism for constructing three-dimensional large scale spatial organizations for applications in areas such as cartography and land use studies, photo interpretation for reconnaissance and surveillance, and geological modeling for resource analysis. It focuses on the representation and utilization of map information as a knowledge source for photo-interpretation, in particular, the description of a highly detailed, large scale geographic area: Washington D.C. Methods of data acquisition, query specification and geometric operations on map data are discussed. These ideas have been implemented into a working map database system, CONCEPTMAP, as a component of MAPS: (Map Assisted Photo-interpretation System), our ongoing research in interactive photo-interpretation work stations.

[McKeown 82b] McKeown, D.M. and J.L. Denlinger.

Graphical Tools for Interactive Image Interpretation.

Computer Graphics 16(3):189-198, July, 1982.

This paper describes BROWSE, an interactive raster image display facility which is a major component of a larger integrated Map Assisted Photointerpretation System (MAPS), being developed as a prototype interactive aid for photointerpretation. Application areas for this research include image cartography, land use studies and reconnaissance, as well as image database organization, storage, and retrieval.

BROWSE is a window-oriented display manager which supports raster image display, overlay of graphical data such as map descriptions and image processing segmentations, and the specification and generation of 3D shaded surface models. Digitized imagery from black and white and color aerial mapping photographs is displayed by BROWSE at multiple levels of resolution and allows for dynamic positioning, zooming, expansion or shrinking of the image window. Map data represented as vectors and polygons can be superimposed on the imagery through image-to-map registration. Access to collateral map databases and terrain models may be accomplished using the BROWSE graphical interface. Finally, the window representation gives a convenient communication mechanism for passing image fragments to image interpretation programs, which generally run as separate processes. The results of such processing can be returned to BROWSE for further processing by the user.

We will discuss the rationale behind the design of BROWSE as well as its application to domains including aerial photointerpretation and 3D cartography.

[McKeown 83] McKeown, D.M., Jr.

Maps: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data.

In Lee S. Baumann, Editor, *14th Proceedings of the DARPA Image Understanding Workshop*, Science Applications, Inc., MacLean, VA., June, 1983.

Reprint no. AD-POO1 199. Also available as CMU-CSD technical report CMU-CS-83-136.

This paper presents the system description and organization of MAPS, the Map Assisted Photo interpretation System. MAPS is a large integrated database system containing high resolution aerial photographs, digitized maps and other cartographic products, combined with detailed 3D descriptions of man-made and natural features in the Washington, D.C. area. Applications of the Maps system in the areas of map-guided image segmentation, rule-based systems for image interpretation, and 3D scene generation are discussed. A classification of image database systems into three models is also presented. These models are the Image Database (ID) Model, the Map Picture Database (MPD) Model and the Image/Map Database (IMD) Model.

[McKeown 84a] McKeown, D.M., Jr.

Digital Cartography and Photo Interpretation from a Database Viewpoint,

In Gargarin, G. and E. Golembe, *New Applications of Data Bases*, Pages 19-42. Academic Press, 1984.

This paper gives an overview of database issues in digital cartography and aerial photointerpretation. A classification of database systems based on the method of data acquisition and underlying spatial representation is described. We also present a brief overview of MAPS, the Map Assisted Photointerpretation System. MAPS is a large integrated database system containing high resolution aerial

photographs, digitized maps and other cartographic products, combined with detailed 3D descriptions of man-made and natural features in the Washington D.C. area.

[McKeown 84b] McKeown, D.M., Jr.

Knowledge-Based Aerial Photo Interpretation.

Photogrammetria, Journal of the International Society for Photogrammetry and Remote Sensing 39:91-123, March, 1984.

Special issue on pattern recognition.

This paper presents an overview of work in two areas that are crucial towards the development of automated tools for aerial photointerpretation: large-scale spatial databases, and rule-based systems for photointerpretation. First, we present a description of models for spatial database systems and outline requirements for database support for knowledge-based photointerpretation. Next we present a brief description of the organization of MAPS, the Map-Assisted Photointerpretation System. MAPS is a large integrated database system containing high-resolution aerial photographs, terrain, digitized maps and other cartographic products, combined with detailed 3D descriptions of man-made and natural features in the Washington D.C. area. Finally, we discuss recent work in the area of rule-based systems for photointerpretation. The system, SPAM, consists of three major components, an image/map database, a collection of image processing tools, and a rule-based system whose domain of expertise is commercial airports.

[McKeown 84c] McKeown, D.M., and J.L. Denlinger.

Map-Guided Feature Extraction From Aerial Imagery.

In *Proceedings of the Second IEEE Computer Society Workshop on Computer Vision: Representation and Control*, IEEE, May, 1984.

In this paper we discuss the use of map descriptions to guide the extraction of man-made and natural features from aerial imagery. An approach to image analysis using a region-based segmentation system is described. This segmentation system has been used to search a database of images that are in correspondence with a geodetic map to find occurrences of known buildings, roads, and natural features. The map predicts the area of uncertainty caused by errors in the image to map correspondence. The segmentation process then searches for image regions that satisfy 2-dimensional shape and intensity criteria. If no initial region is found, the process attempts to merge together those regions that may satisfy these criteria. Several detailed examples of the segmentation process are given.

[McKeown 84d] McKeown, D.M., Jr.

Spatial Database Research at CMU.

IEEE 1984 Proceedings of Trends and Applications (CH2053-7/84/0000/0319):319-323, July, 1984.

This paper gives a brief overview of current research at Carnegie-Mellon University in the area of spatial database systems for digital cartography and aerial photointerpretation. A brief overview of MAPS, the Map-Assisted Photointerpretation System, is presented. MAPS is a large integrated database system containing high resolution aerial photographs, digitized maps, and other cartographic products, combined with detailed 3D descriptions of man-made and natural features in the Washington D.C. area.

[McKeown 84e] McKeown, D.M., and G.E. Lukes.

Digital Mapping and Image Understanding.

In *Archives of the XVth Congress on Photogrammetry and Remote Sensing*, Pages 690-697.

International Society for Photogrammetry and Remote Sensing, June, 1984.

Rio de Janeiro, Brazil.

Emerging requirements associated with digital mapping pose a broad set of challenging problems in image understanding research. Currently several leading research centers are pursuing the development of new techniques for automated feature extraction; for example, road tracking, urban scene generation, and edge-based stereo compilation. Concepts for map-guided scene analysis are being defined which will lead to further work in automated techniques for spatial database validation, revision and intensification.

This paper seeks to describe on-going activity in this field and suggest areas for future research. Research problems range from the organization of large-scale digital image/map databases for tasks such as screening and assessment, to structuring spatial knowledge for image analysis tasks, and the development of specialized "expert" analysis components and their integration into automated systems. Significantly, prototype image analysis workstations have been configured for both film-based and digital image exploitation which interface conventional image analysts and extracted spatial data in computer-assisted systems. However, the state-of-the-art research capabilities are fragile, and successful concept demonstrations require thoughtful analysis from both the mapping and image understanding communities.

[Ohta 83]

Ohta, Y. and T. Kanade.

Stereo by Intra- and Inter-scanline Search Using Dynamic Programming.

Technical Report CMU-CS-83-162, Carnegie Mellon University, Computer Science Department,

October, 1983.

This paper presents a stereo matching algorithm using the dynamic programming technique. the stereo matching problem, that is, obtaining a correspondence between right and left images, can be cast as a search problem. When a pair of stereo images is rectified, pairs of corresponding points can be searched for within the same scanlines. We call this search *intra-scanline* search. This intra-scanline search can be treated as the problem of finding a matching path on a two-dimensional (2D) search plane whose axes are the right and left scanlines. Vertically connected edges in the images provide consistency constraints across the 2D search planes. *Inter-scanline* search in a three-dimensional (3D) search space, which is a stack of the 2D search planes, is needed to utilize this constraint.

Our stereo matching algorithm uses edge-delimited intervals as elements to be matched, and employs the above mentioned two searches: one is inter-scanline search for possible correspondences of connected edges in right and left images and the other is intra-scanline search for correspondences of edge-delimited intervals on each scanline pair. Dynamic programming is used for both searches which proceed simultaneously: the former supplies the consistency constraint to the latter while the latter supplies the matching score to the former. An interval-based similarity metric is used to compute the score.

The algorithm has been tested with different types of images including urban aerial images, synthesized images, and block scenes, and its computational requirement has been discussed.

[Shafer 82a]

Shafer, S.A., and T. Kanade.

Gradient Space Under Orthography and Perspective.

In *Proceedings of the IEEE Workshop on Computer Vision: Representation and Control*, Pages 22. IEEE Computer Society, P.O. Box 80452, Los Angeles, CA 90080, August, 1982.

Also available as CMU-CSD Technical Report CMU-CS-82-123.

Mackworth's gradient space has proven to be a useful for image understanding. However, descriptions of its important properties have been somewhat scattered in the literature.

This paper develops and summarizes the fundamental properties of the gradient space under orthography and perspective, and for curved surfaces. While largely a recounting of previously published results, there are a number of new observations, particularly concerning the gradient space and perspective projection. In addition, the definition and use of vector gradients as well as surface gradients provides concise notation for several results.

The properties explored in the paper include the orthographic and perspective projections themselves; the definition of gradients; the gradient space consequences of vectors (edges) belonging to one or more surfaces, and of several vectors being contained on a single surface; and the relationships between vanishing points, vanishing lines, and the gradient space.

The paper is intended as a study guide for learning about the gradient space, as well as a reference for researchers working with gradient space.

[Shafer 82b]

Shafer, S., and T. Kanade.

Using Shadows in Finding Surface Orientation.

In *13th Proceedings of the DARPA Image Understanding Workshop*, Pages 61. Science Applications, Inc., MacLean, VA, September, 1982.

Also available as CMU-CSD Technical Report CMU-CS-82-100.

Given a line drawing from an image with shadow regions identified, the shapes of the shadows can be used to generate constraints on the orientations of the surfaces involved. This paper describes the theory which governs those constraints under orthography.

A 'Basic Shadow Problem' is first posed, in which there is a single light source, and a single surface casts a shadow on another (background) surface. There are six parameters to determine: the orientation (2 parameters) for each surface, and the direction of the vector (2 parameters) pointing at the light source. If some set of three of these are given in advance, the remaining three can then be determined geometrically. The solution method consists of identifying 'illumination surfaces' consisting of illumination vectors, assigning Huffman-Clowes line labels to their edges, and applying the corresponding constraints in gradient space.

The analysis is extended to shadows cast by polyhedra and curved surfaces. In both cases, the constraints provided by shadows can be analyzed in a manner analogous to the Basic Shadow Problem. When the shadow falls upon a polyhedron or curved surface, similar techniques apply. The consequences of varying the position and number of light sources are also discussed. Finally, some methods are presented for combining shadow geometry with other gradient space techniques for three-dimensional shape inference.

[Shafer 83a]

Shafer, S. A., and T. Kanade.

The Theory of Straight Homogenous Generalized Cylinders and A Taxonomy of Generalized Cylinders.

In Lee S. Baumann, Editor, *14th Proceedings of the DARPA Image Understanding Workshop*, Science Applications, Inc., MacLean, VA, January, 1983.

Reprint no. AD-POO1 209. Also available as CMU-CSD technical report CMU-CS-83-105.

In recent years, Binford's generalized cylinders have become a commonly used shape representation scheme in computer vision. However, research involving generalized cylinders has been hampered by a lack of analytical results at all levels, even including a lack of a precise definition of these shapes.

In this paper, a definition is presented for Generalized Cylinders and for several subclasses. Straight Generalized Cylinders, with a linear axis, are important because the natural object-centered coordinates are not curved. The bulk of the paper is concerned with Straight Homogenous Generalized Cylinders, in which the cross-sections have constant shape but vary in size.

The results begin with deriving formulae for points and surface normals for these shapes. Theorems are presented concerning the conditions under which multiple descriptions can exist for a single solid shape. Then, projections, contour generators, shadow lines, and surface normals are analyzed for some subclasses of shapes. The strongest results are obtained for solids of revolution (which we have named Right Circular SHGCs), for which several closed-form methods for analyzing images are presented.

[Shafer 83b]

Shafer, S.A.

Shadow Geometry and Occluding Contours of Generalized Cylinders.

PhD thesis, Carnegie-Mellon University, May, 1983.

Also available as CMU-CSD technical report CMU-CS-83-131.

Given a line drawing from an image with shadow regions identified, the shapes of the shadows can be used to generate constraints on the orientations of the surfaces involved. This thesis describes the theory which governs those constraints and shows how it can be applied to polyhedra and certain types of generalized cylinders.

[Shafer 84]

Shafer, S.A.

Optical Phenomena in Computer Vision.

In *Proceedings of the Canadian Society for Computational Studies of Intelligence Conference*, London, Ontario, Pages 1-34. May, 1984.

Computer vision programs are based on some kind of model of the optical world, in addition to whatever significance they may have in terms of human vision, algorithms, architectures, etc. There is a school of research that addresses this aspect of computer vision directly, by developing mathematical models of the optics and geometry of image formation and applying these models in image understanding algorithms. In this paper, we examine the optical phenomena that have been analyzed in computer vision and suggest several topics for future research.

The three topics that have received the most attention are shading (and glossiness), color, and shadows. Shape-from-shading research, while producing many interesting algorithms and research results, has primarily been based on very simplified models of glossiness. Since realistic gloss models exist within the optics community, we can expect improved computer vision algorithms in the future. Color work in the past has similarly concentrated on developing sophisticated algorithms

for exploiting very simple color models, but a more realistic analysis technique has recently been proposed. Shadows have been used by a number of people for simple analysis such as locating buildings in aerial photographs, and a more complex theory already exists that relates surface orientations to shapes of shadows in the image.

A number of problems plague this kind of research, however, including the current inability to model real complexities of illumination and reflection, and the nagging feeling that humans don't seem to rely upon very quantitative analysis of optical properties of materials and illumination. These questions are also addressed.

[Smith 84]

Smith, D., and T. Kanade.

Autonomous Scene Descriptions with Range Imagery.

In Lee S. Baumann, Editor, *Proceedings of the DARPA Image Understanding Workshop*, Science Applications, Inc., MacLean, VA, October, 1984.

This paper presents a program to produce object-centered three-dimensional descriptions starting from point-wise 3D range data obtained by a light-stripe rangefinder. A careful geometrical analysis shows that contours which appear in light-stripe range images can be classified into eight types, each with different characteristics in occluding vs. occluded and different camera/illuminator relationships. Starting with detecting these contours in the iconic range image, the descriptions are generated moving up the hierarchy of contour, surface, object, to scene. We use conical and cylindrical surfaces as primitives. In this process, we exploit the fact that coherent relationships, such as symmetry, collinearity, and being coaxial, which are present among lower-level elements in the hierarchy allow us to hypothesize upper-level elements. The resultant descriptions are used for matching and recognizing objects. The analysis program has been applied to complex scenes containing cups, pans, and toy shovels.

[Specker 83]

Specker, P.

A Post-Processing Algorithm for Time Domain Pitch Trackers.

Technical Report CMU-CS-83-104, Carnegie Mellon University, Computer Science Department,

January, 1983.

This paper describes a powerful post-processing algorithm for time-domain pitch trackers. On two successive passes, the post-processing algorithm eliminates errors produced during a first pass by a time-domain pitch tracker. During the second pass, incorrect pitch values are detected as *outliers* by computing the distribution of values over a sliding 80 msec window. During the third pass (based on artificial intelligence techniques), remaining pitch pulses are used as anchor points to reconstruct the pitch train from the original waveform. The algorithm produced a decrease in the error rate from 21% obtained with the original time domain pitch tracker to 2% for isolated words and sentences produced in an office environment by 3 male and 3 female talkers. In a noisy computer room errors decreased from 52% to 2.9% for the same stimuli produced by 2 male talkers. The algorithm is efficient, accurate, and resistant to noise. The fundamental frequency microstructure is tracked sufficiently well to be used in extracting phonetic features in a feature-based recognition system.

[Thorpe 84]

Thorpe, C.E.

Fido: Vision and Navigation for a Robot Rover.

PhD thesis, Carnegie Mellon University, Computer Science Department, December, 1984.

Fido is a vision and navigation system for a mobile robot. Using only stereo vision, **Fido** is capable of guiding a robot through clutter to reach a goal. **Fido** starts with no preloaded map of the world, and needs no special preparation of the environment; it works in the real world rather than in a toy environment. This thesis describes **Fido**'s major contributions in path relaxation, interest operators, constraints, and parallel decomposition. The thesis concludes with a recommendation for hardware and software improvements for the current vehicles, and for the design of future mobile robot systems.

4. Machine Intelligence

To significantly improve their performance, artificial intelligence systems need the ability to acquire, represent, organize, and effectively utilize large amounts of knowledge. Our work in this area displays two facets:

- Knowledge Representation addresses the automated acquisition of knowledge (factual, episodic, procedural, heuristic), its internal organization, and the inference processes that access the resultant knowledge bases to solve interesting problems.
- Knowledge Engineering addresses issues of the construction of computer systems that can solve complex real-world problems by capturing and codifying human expertise.

This chapter reports on recent achievements in these broad areas of artificial intelligence research at Carnegie Mellon University.

4.1. Knowledge Representation

Our principle objective in knowledge representation research is to explore methods that enable automated systems to perform effectively in ill-structured problem domains, though the systems may have only incomplete or unreliable knowledge. Our central strategy is to investigate various methods in the context of well-defined problems (rather than in vacuo) such as robot planning, algorithm design, massively-parallel architectures, and counterplanning scenarios such as playing chess. These task areas serve merely as vehicles for discovering and validating general problem-solving methods. Creating and refining the methods advances artificial intelligence.

During our investigations we focused on developing substantial *cognitive architectures* that can learn, solve problems, plan, and reason across varied task domains. The NETL and Boltzmann projects examine how to construct massively parallel associative memories, realizable (in principle) directly on the hardware. The Soar architecture provides a universal problem-solving engine that unifies all the weak methods (heuristic search, means-ends analysis, etc.) and can improve its performance with experience by chunking search-control knowledge. Prodigy is a learning apprentice system: a general problem solver that can take instruction, explore its environment, and analyze its own behavior to achieve expert-level behavior within a large problem class, chiefly complex planning domains. Additionally, we are developing and refining more well-defined techniques, such as transformational and derivational analogy, application coefficients for heuristic evaluation functions, algorithm design methods, and knowledge acquisition techniques for expert systems. The following sections discuss specific projects in greater detail.

4.1.1. Machine Learning

We view machine learning as an integral part of planning and problem-solving where the learner must acquire both problem-solving expertise in general or specific domains and more transitional facts and general concepts. Our main accomplishments in the reporting period are summarized below:

- We developed a problem-solving method called *transformational analogy*, capable of exploiting past experience by transforming solutions of related problems into the solution of the new problem at hand [Carbonell 82]. The primary benefit was a qualitative speedup in problem-solving as the system's relevant experience in related areas could be transferred and exploited, rather than rederived at much cost in combinatorial search. Secondary benefits included the automated induction of general plans from multiple analogically-related solutions to similar problems. In essence, transformational analogy provided the grist for an inductive generalization engine, one capable of abstracting general plans from the shared aspects of multiple similar-instance solutions. These general plans can be applied directly with little need for back-to-basics problem-solving, or even analogical transfer. Of course, for qualitatively different classes of problems, the general problem-solver is still required. In such a manner, our Aries problem-solving systems can learn from experience—and can solve, effectively and quickly, complex problems in familiar areas, but must resort to the much slower step-by-step exploration of the problem space for less familiar areas.
- Subsequently we developed the *derivational analogy* problem-solving method to augment the earlier transformational analogy paradigm. This method transfers strategies successful in similar past problem-solving instances solve new, possibly more complex problems and serves as a basis for inductive generalization of new planning methods. Thus, derivational analogy serves to learn not just general plans, but *planning strategies*. Those which are transferred from similar problem-solving episodes are the strategies themselves and the justification of why they worked (or why they failed to work). Derivational analogy is a central technique incorporated into the Prodigy and World Modellers projects discussed below.
- Recently we designed and implemented the first version of the *world modellers robotic simulator* [Hood 82]. The system's three-dimensional, Newtonian physics simulator enables exploration and verification of robotic planning and learning techniques without requiring a "real" robot—and all the difficult vision and locomotion problems that entails. Our work investigates *fractioning*, *causal attribution* and *T-macro* techniques for focused learning in the reactive simulation environment. The ability to run goal-directed experiments in a reactive environment, to reason about their effects, to formulate hypotheses explaining why reality deviates from expectations, and to design further experiments seeking to confirm, augment or falsify these hypotheses in a rational symbolic manner is a crucial component of learning heretofore ignored by most of the symbolic-reasoning/AI machine-learning community.
- The Soar and Boltzmann projects contain significant machine learning components in addition to their central concern with building a general cognitive architecture.

4.1.2. Adversarial Problem Solving

Representing Uncertain Knowledge with Continuous Functions

An important aspect of intelligence is arriving at conclusions in the absence of complete knowledge or in the presence of uncertain future events. Uncertain knowledge must be represented continuously and united effectively with symbolic knowledge. The advent of application coefficients in non-linear heuristic evaluation functions has made the integration of uncertain knowledge with symbolic knowledge possible [Berliner 81]. We have successfully developed two direct descendants of this new technology: Iago, our world-champion Othello-playing program [Rosenbloom 81], and BKG, our equally successful backgammon program.

The performance of BKG, which defeated the World Backgammon Champion, improved greatly when we changed its knowledge representation from the form of rules to continuous functions [Ackley 83]. A question coming out of this work is how such continuous knowledge can be communicated to the outside world, since we must convert to a discrete representation and find words that match the concepts being expressed. The QBKG system [Berliner 82] was able to compare a user-selected backgammon move to BKG's own best choice, then give a detailed analysis of the pros and cons of each move. QBKG has made a considerable impression on the expert system field, where little had been done with continuous knowledge representations prior to our research. We have distributed over 2000 copies of the QBKG System report in response to requests.

The B* search is still demonstrably the best selective search for adversary problems. In his work with the B* algorithm, Palay discovered that substituting distribution functions for ranges in representing node values significantly improved B*'s performance [Palay 83]. He showed this in a series of simulations using standard chess tactics problems. However, manipulating distributions and searching distribution data remain extremely time consuming. Implementing a practical system awaits hardware that can efficiently accomplish these tasks.

Using Chunking to Solve Chess Pawn Endgames

When using simple representations in evaluating competitive situations like chess, the system must employ search techniques to find consequences undiscerned by the representation used. For example, when an evaluation function examines only the roles of single pieces, it misses much of what could be understood by examining interactions among related pieces. The latter strategy involves what psychologists call "chunking" and we use that term when describing our work.

Examining a chess board as a set of piece-sets rather than as a set of single pieces provides a more profound representation of the situation. Campbell and Berliner developed such a representation in the chess pawn endgame they studied [Campbell 83]. They devised a method of chunking the various possible configurations and the resulting Chunker program [Berliner 83] played certain positions in its domain 10^{13} times faster than

the best programs using the old representation. Chunker found two mistakes in the literature relating to this endgame, though this particular game has been the subject of study for over 300 years. Chunker, now a complete master of its domain, makes no mistakes in assessing any position in this endgame. Campbell's and Berliner's approach saves many orders of magnitude of search. They have expanded their work to include the entire domain of chess pawn endgames.

Massive Search using Constraint Satisfaction

Berliner developed a program named Superpuzz that solves card game puzzles. Though equipped with an excellent knowledge function and a rapid search algorithm, Superpuzz initially failed to outperform its creator. After Berliner added constraint satisfaction methods that ruled out branches of the search determined not to contain any possible solutions, Superpuzz always outperformed Berliner and it became highly improbable a human would find better solutions.

Berliner studied three variants of the puzzle along with several different search techniques and evaluation functions [Berliner 84]. He found that the constraint satisfaction technique helped selective (best-first) searches with knowledgeable evaluation functions significantly more than it aided other searches. He also discovered that a method he calls "adventurousness," which gives greater credit to achievement than debit for effort expended effort, decreases time-to-solution in all search paradigms.

4.1.3. Massively Parallel Cognitive Architectures

A critical problem in knowledge representation is storing and accessing immense amounts of real-world knowledge. Our research in this area led to the design and development of the NETL system, a specialized, massively parallel machine architecture. Essentially, NETL is a semantic network data structure with a very simple hardware device representing each network node and link. Its design allows it to solve search and inference problems that had proven intractable to the traditional serial heuristic approaches.

During the 1981-84 period, our NETL research emphasized two themes: developing a practical approach to implementing NETL in hardware and developing a solid formal understanding of the knowledge representations it uses, particularly in the area of default reasoning and exceptions. The first theme led to several preliminary designs for a million-element NETL machine and inspired the Connection Machine work at MIT. The second theme led to a formal mathematical theory of inheritance [Touretzky 84] that has been influential in the knowledge representation community.

In time, we shifted our emphasis away from NETL, where knowledge is tied to a specific hardware element, to the Boltzmann Machine architecture [Fahlman 83]. The Boltzmann architecture has three advantages: first, it can extract new knowledge from examples, rather than requiring each new fact to be input by hand; second,

it employs a distributed representation that is inherently fault-tolerant and therefore better suited for very large-scale VLSI implementations; third, a Boltzmann machine resembles a neural network and may provide us with some useful models for understanding information processing in the brain. We continue to pursue the Boltzmann approach vigorously and it has given rise to an entire family of related architectures.

4.1.4. A Universal Problem-Solving Architecture

Within this contract period we developed an architecture for general intelligence, called Soar [Laird 84a], that unites many of the important ideas and mechanisms concerning production systems and problem spaces we have explored for twenty years. Soar adopts a uniform view that all cognitive activity can be represented as search in a problem space [Newell 80]—whether problem-solving or routine action. Thus, Soar is a problem-solving architecture whose primitive actions are selecting problem spaces, states, and operators, and applying the operators by selecting another implementation problem space. We realized Soar within a production system (a slight modification of OLC [Forgy 84]) that uses both search control and the implementation of primitive operators. From the expert systems point-of-view, Soar provides a production system framework that is hierarchically organized in terms of problem spaces, instead of a single flat set of rules.

The Soar framework embodies several other important discoveries and integrations, among them the principle of *universal subgoalting* [Laird 84b]. Soar, as with many other systems, focuses its attention by the difficulties it encounters, but because the architecture is cast at the problem-solving level, these difficulties (called *impasses*) can all be detected directly by the architecture. Thus the architecture itself can detect the need for a subgoal, develop its initial characterization, and set it up. This subgoalting method has far-reaching consequences. For example, it permits Soar to recognize instantly when any goal at any level in the goal stack has succeeded or failed, and to jump back immediately to continue from that point. Impasses become the fundamental basis for detecting conflict resolution needs. The production system no longer has a conflict-resolution stage, but instead lets all productions fire in parallel.

The study of weak methods (generate and test, means-ends analysis, etc.) has been a significant part of our research effort since the late 1960s when it became clear how important they are in all problem-solving systems. Soar represents an advance in implementing weak methods. In effect, when provided with the essential knowledge on which a particular weak method is based, Soar behaves according to that method. This was originally called the universal weak method [Laird 83], although it now seems preferable to term this simply an implicit mode of methods representation.

Each weak method of course builds on and exploits some explicit knowledge about the task structure. A key issue is whether a production system needs additional procedural knowledge to exploit the task knowledge.

Standard programming systems need additional procedural knowledge, Soar doesn't. This occurs because: (1) all the weak methods turn out to be search methods fundamentally and Soar embodies heuristic search in its very architecture; (2) Soar's use of production systems to represent search control provides a decomposition that permits each bit of heuristic knowledge to be given separately; and (3) the weak methods are themselves quite simple, being in effect the "obvious" ways of exploiting a small amount of knowledge about the task.

Soar also learns as it solves problems. Rosenbloom and Newell researched how human performance at problem-solving improves with practice [Rosenbloom 83]. They developed the theory that the memory organization method called *chunking* in Cognitive Psychology provides a sufficient mechanism for improved performance. Then they implemented the chunking theory in a series of production system architectures (XAPS1 and XAPS2 [Rosenbloom 82]), and demonstrated that creating chunks (as productions) not only permits continuous improvement, but follows the same quantitative law of improvement as humans do (the so-called *power law of practice* [Rosenbloom 82]). In late 1984 they transferred their ideas into Soar's problem-solving architecture; now Soar chunks all of its goal results. If Soar ever needs to solve a previously solved goal, it will be able to do so directly without re-solving the problem. The chunking mechanism seemed at first to be one of pure practice, but now appears to be a general mechanism that transfers to new situations. They are now exploring the hypothesis of whether it is sufficient for all types of learning.

The Soar architecture unifies many of our research advances made during the 1981-84 period and preceding research contracts as well. We prepared a major demonstration of Soar by developing R1-Soar [Rosenbloom 84] that accomplished the same task as R1, the 3300-rule VAX-configuring system [McDermott 82], yet included only 25% of R1's functionality. R1-Soar is an example of how a general problem-solving system can be transformed into a knowledge-intensive system by the addition of search control. It demonstrates that chunking is sufficient to let the R1-Soar system develop this same search control from its own experience, supporting the basic proposition that Soar learns about whatever task it tackles.

4.2. Knowledge Engineering

4.2.1. An Algorithm Design Assistant

Kant and Newell have been exploring the domain of algorithm design as an instance of problem-solving and expert behavior. They assert that algorithm design demands much more intellectually in comparison to most expert system tasks, that it combines problem-solving and expert behavior in important ways, and that it may offer a new approach to program synthesis, in that the output of an algorithm design system provides a new knowledge structure for specifying the program synthesis task. During the contract period we moved

from an initial formulation of the task and the selection of a specific domain, geometric algorithms (such as designing an algorithm for finding the convex-hull planar point-set). We then developed the basic representation and methods (by analyzing in detail the behavior of human experts), to the design and implementation of a system, Designer, for doing the task [Kant 84].

We tackled a wide range of issues in the Designer project, including a representation for partially specified algorithms (as a data-flow scheme with associated assertions) and the symbolic execution of these partial algorithms, in a more general framework than has usually been attempted in programming systems schemes for symbolic execution. We have advanced it to a point where it can execute many (partial) algorithms and can design some simple ones, and are nearing our initial benchmark task, designing the convex-hull algorithm.

Designer has grown into a very large system and incorporates both a frame system and a production system. Due to the successful implementation of Soar, with its extremely clean structure, and to some personnel changes, we are in the midst of reimplementing Designer in Soar.

4.2.2. Architectures for Fast, Intelligent Search

In order to speed systematic analysis in adversarial search (e.g. minimax with alpha-beta pruning), we investigated new methods for parallel decomposition and for performing the search at the silicon level. We chose chess as the test domain for this emerging general technology, and to that end we designed and fabricated custom chips for different aspects of the game—some particular to chess, others more general to decision-making that requires systematic search. These latter chips operate by first downloading a programmable decision function and later executing the function at blinding speed.

4.2.3. Massive Search Systems

Intelligent systems can be viewed as working in a space of directly available knowledge, on one dimension, while searching to obtain more knowledge in a second dimension. Any specific system employs some combination of these two activities. Humans occupy the high-knowledge, low-search part of the space. In general, expert systems designers have explored that area, though all regions of the space are of interest. Game playing programs, especially for chess, have become a mainstay in the high-search, low-knowledge area, demonstrating that improving the search was the most effective way to attain high performance. We have been exploring this path for well over ten years. Within the scope of the present contract, we have made an important breakthrough with a demonstration of a chess system (Hitech) that is substantially better than any existing system [Ebeling 84].

Hitech is a high-search, low-knowledge system, not a pure mass-search system, though the knowledge it contains is extremely important. Its performance has improved significantly as we've added knowledge, and we have not yet found the system's performance limits. What Hitech does show is that solutions in the high-search range are finally effective in achieving excellent performance in a task in which humans also achieve excellence, but by a radically different combination of search and knowledge.

Hitech consists of a workstation host plus a special machine that searches about 175,000 positions per second. The machine is organized as a set of processors on a common bus and consists of a move generator (64 copies of a VLSI chip specially designed via the MOSIS facility), a set of evaluation modules, and a memory unit that remembers previously explored positions and their values. The whole system operates by downloading the specific evaluations to be made in the current position, then searching up to 30 million positions to find the preferred move. Hitech contains all techniques that have proved valuable in our decade of research: iterative deepening, adaptive coefficients (the SNAC procedure, developed under this research program in building BKG, our world-champion backgammon program), recalling large numbers of prior search positions, etc. Hitech won the recent ACM chess tournament against all other programs, and in play against humans earned an official rating of about 2070, well into the Master range.

4.2.4. Prodigy: A Learning Apprentice

In 1984 we embarked upon the Prodigy learning apprentice project. The system models an insightful student, one capable of general problem solving but with little initial knowledge of any given domain. Prodigy gradually builds up expertise through:

- Introspective assessment and analytical improvement of its planning processes
- Accepting human instruction on domain-specific information, including suggestions for making the new information operational
- Formulating experiments (either questions to the teacher or procedures to be carried out in its environment)
- Analytic generalization where a strong domain theory is available, or empirical generalization when domain knowledge for provably correct generalizations is lacking

Goal regression and weakest-precondition analysis are the central tools for analytic generalization. So far we have conducted experiments to gather missing information required for weakest-precondition analysis, and to establish correct analogical mappings.

4.2.5. Speech Recognition

Our research goal in speech recognition is to generate and verify general techniques for developing high-performance, special-purpose, knowledge-based systems. The speech domain allows us to formulate and validate some of these techniques in a practical task. The following paragraphs discuss the two lower-level aspects of this research. At the uppermost level, this work blends into the User Interface Research discussed in Chapter 7. See section 7.5 for a discussion of the voice message system.

Acoustic/Phonetic Recognition

Two challenges in computer speech recognition are (a) performing speaker-independent recognition, and (b) discriminating among speech sounds that are acoustically similar, such as [b] and [d]. Prior to 1981, speech recognition systems were speaker-dependent and required each new speaker to train the system to his or her voice. The systems made many errors when the vocabulary items were acoustically similar.

A main goal of CMU speech research between 1980 and 1982 was to demonstrate the ability to perform speaker-independent recognition of confusable words [Lasry 84]. In order to make the problem tractable, we decided to recognize letters of the English alphabet spoken in isolation. This task domain provides a useful and well defined vocabulary that contains many fine phonetic distinctions, as in the set B, D, E, P, T, G, V, Z, C [Waibel 81].

We decided to build an expert system to model the performance of an expert spectrogram reader. The research involved studying speech spectrograms of letters spoken by many speakers, to discover the recognition features needed for each letter; developing feature-measuring algorithms to quantify the perceptually relevant features; and using a multivariate classifier to combine the feature values for letter choice decisions. Our research produced a speaker-independent, isolated letter recognition system called Feature [Stern 83]. Feature performed at levels of accuracy significantly better than any previous system. In speaker-independent mode, letters were correctly recognized about 90% of the time. When the system was allowed to automatically learn about the speech patterns of each user, performance rose to 95%.

Word Recognition for Large Vocabularies

The Word Recognition project at CMU has investigated two major topics: efficient search algorithms and automatic generation of lexical representations. Our goal is to develop a word recognition technique that deals flexibly with the requirements of large vocabulary recognition.

Controlling the size of the search space becomes a critical problem as the size of a recognition vocabulary increases. Research on search techniques has focused on identifying effective constraints. Rudnick, together with Lehmann, has developed a search algorithm that makes use of several sources of constraint to improve search efficiency [Rudnick 82]. Constraint is provided by using two, independently generated speech encod-

ings: a phonetic lattice and a coarse-class lattice. The coarse lattice is used to identify syllable nuclei and thereby identify potential word locations. The coarse-class lattice is also used to provide a consistency check. Phonetic and coarse lattice must agree on their description of the input in order for the search to proceed. Together these constraints produce approximately a 40-fold reduction in search for a typical utterance, as well as (somewhat surprisingly) an increase in recognition accuracy [Waibel 82].

Traditional recognition schemes require a substantial effort to construct an accurate representation through training. Such techniques become less feasible as vocabulary size increases. The goal of our research in this area is to develop techniques for automatically generating reliable and complete descriptions of lexical items. Rudnicky has developed a system for the automatic transformation of lexical baseforms ("dictionary pronunciation") into lexical networks using declaratively specified phonological rules. Hand-labeled speech was analyzed to produce these phonological rules.

4.2.6. SPAM: Rule-Based Systems for Aerial Photointerpretation

McKeown has begun to explore and develop the use of rule-based systems for interpreting complex, high resolution aerial photography [McKeown 83]. SPAM, a System for Photointerpretation of Airports using MAPS (see Chapter 3), interprets scenes of the National Airport in Washington, D.C. The system uses map descriptions of the airport layout, and tools for spatial reasoning about size, shape, and position of various airport features. We focus on building a knowledge base to control image processing primitives and to guide the image interpretation process. Our long term goal is to develop systems that maintain a world database of previous events. The world database's expert level knowledge will be able to predict areas for fruitful analysis and will integrate the analysis' results into a coherent model.

Combining map knowledge and a rule-based system for scene analysis is a new approach to model-based vision that allows us to decouple the task domain from the low-level image processing tools and to integrate spatial, general model, and site-specific knowledge within a single framework. Using local evidence, it is possible to make weak interpretations about the mapping of image segments to a model, and to trigger continually more refined segmentation hypotheses. Interpretation is performed by recognizing that only a small subset of the large number of hypotheses are mutually consistent. Inconsistencies can be detected by analysis of geometric relationships between local features and the application of world knowledge, such as the typical length of runways, size of hangars and maintenance buildings, and their relative spatial organization within a general airport scene.

Currently SPAM can extract and identify some runways, taxiways, grassy areas, and buildings in several images of the National Airport. It cannot yet, however, recognize a complete airport scene that satisfies its

MACHINE INTELLIGENCE

internal map model. Our research continues in several areas including reliable low-level feature extraction from the imagery, and the design and implementation of effective recognition strategies using the rule-based approach. We believe that integrating map knowledge, image processing tools, and rule-based control and recognition strategies will be prove a powerful computational organization for automated image analysis.

4.3. Bibliography

- [Ackley 83] Ackley, D.H. and H.J. Berliner.
The QBKG system: Knowledge representation for producing and explaining judgements.
 Technical Report CMU-CS-83-116, Carnegie-Mellon University, Computer Science
 Department,
 March, 1983.
 The QBKG system plays backgammon and produces critical analyses of possible moves for a wide variety of backgammon positions, using a hierarchically structured, non-discrete form of knowledge representation. The largely non-searching control structure emphasizes *judgemental* processes at the expense of reasoning processes, meaning that the system's behavior is determined by the estimated usefulness of its immediate actions rather than upon hypothesized longer-term results such as would be produced by a tree-searching algorithm. This report describes some of the principles by which knowledge can be represented so as to facilitate high-quality judgements in a domain, discusses issues arising from the need to be able to explain how a particular judgement was reached, and argues that sophisticated judgemental ability is a critical feature for systems operating in complex, incompletely understood environments.
- [Berliner 81] Berliner, H.J.
Search vs. knowledge: An analysis in the domain of games.
 Technical Report CMU-CS-82-104, Carnegie-Mellon University, Computer Science
 Department,
 November, 1981.
 We examine computer games in order to develop concepts of the relative roles of knowledge and search. The paper concentrates on the relation between knowledge applied at leaf nodes of a search and the depth of the search that is being conducted. Each knowledge of an advantage has a projection ability (time to convert to a more permanent advantage) associated with it. The best programs appear to have the longest projection ability knowledge in them. If the application of knowledge forces a single view of a terminal situation, this may at times be very wrong. We consider the advantages of knowledge delivering a range as its output, a method for which some theory exists, but which is as yet unproven.
- [Berliner 82] Berliner, H.J. and D.H. Ackley.
 The QBKG System: Generating Explanations from a Non-Discrete Knowledge Representation.
 In *Proceedings of the National Conference on Artificial Intelligence*, Pages 213-216. The American Association for Artificial Intelligence, August, 1982.
 The QBKG system produces critical analyses of possible moves for a wide variety of backgammon positions using a hierarchically structured, non-discrete form of knowledge representation. This report compares discrete and continuous representations and reasoning systems, addressing issues of competence, robustness, and explainability. The QBKG system is described and demonstrated.

[Berliner 83]

Berliner, H.J. and M. Campbell.

Using Chunking to Solve Chess Pawn Endgames.

Artificial Intelligence 23, April, 1983.

Also presented at the Second International Symposium on Artificial Intelligence and the Game of Chess, Milan, Italy, May, 1983. Also published as CMU-CSD technical report CMU-CS-83-122.

Chunker is a chess program that uses chunked knowledge to achieve success. Its domain is a subset of king and pawn endings in chess that has been studied for over 300 years. Chunker has a large library of chunk instances where each chunk type has a property list and each instance has a set of values for these properties. This allows Chunker to reason about positions that come up in the search that would otherwise have to be handled by means of additional search. Thus the program is able to solve the most difficult problem of its present domain (a problem that would require 45 ply of search and on the order of 10^{13} years of CPU time to be solved by the best of present day chess programs) in 18 ply and one minute of CPU time. Further, Chunker is undoubtedly the world's foremost expert in its domain, and has discovered 2 mistakes in the literature and has been instrumental in discovering a new theorem about the domain that allows the assessing of positions with a new degree of ease and confidence. In this paper we show how the libraries are compiled, how Chunker works, and discuss our plans for extending it to play the whole domain of king and pawn endings.

[Berliner 84]

Berliner, H. and G. Goetsch.

A Quantitative Study of Search Methods and the Effect of Constraint Satisfaction.

Technical Report CMU-CS-84-147, Carnegie-Mellon University, July, 1984.

While certain guidelines for selecting search methods have emerged over the years, and while the success of certain methods has been well documented, there have been few, if any, studies of a variety of search methods as the difficulty of a particular problem changes. This study of a solitaire puzzle that can be incarnated in a variety of problem sizes and hence difficulties, attempts to fill this vacuum. We select four search paradigms: A*, Best-first with a simple evaluation function (BF1), Best-first with a complex evaluation function (BF2), and Depth-First with branch and bound and iterative deepening (DF), to represent the best of a variety of searching methods for non-adversary problems. All methods except BF2 use the same knowledge; i.e. a measure of the minimum number of moves that it would take to win the current situation. DF applies this measure only at maximum depth, while the other two use it for selecting which node to expand next. Each of these methods are tested with and without a constraint satisfaction procedure.

As expected, the most informed search (BF2) does better than the less informed as the problems get progressively more difficult. One important and apparently general result is that constraint satisfaction provides the greatest gain when coupled with the most informed algorithm (BF2). We found it surprising that BF1, which uses the same evaluation function as A* but with a different coefficient, far outperformed A* in terms of work required at about a 5% reduction in the quality of the (otherwise) optimal solution. We conjecture that Best-first searches get their power from an *adventurous coefficient* which we define in the text. Adventurousness can be thought of as a primitive form of planning.

- [Campbell 83] Campbell, M. and H.J. Berliner.
A Chess Program that Chunks.
In *Proceedings of the National Conference on Artificial Intelligence*, Pages 49-53. AAAI,
August, 1983.

Chunker is a chess program that uses chunked knowledge to achieve success. Its domain is a subset of king and pawn endings in chess that has been studied for over 300 years. Chunker has a large library of chunk instances where each chunk type has a property list and each instance has a set of values for these properties. This allows Chunker to reason about positions that come up in the search that would otherwise have to be handled by means of additional search. Thus the program is able to solve the most difficult problem of its present domain (a problem that would require 45 ply of search and on the order of 10^{13} years of CPU time to be solved by the best of present day chess programs) in 18 ply and one minute of CPU time. Further, Chunker is undoubtedly the world's foremost expert in its domain, and has discovered two mistakes in the literature and has been instrumental in discovering a new theorem about the domain that allows the assessing of positions with a new degree of ease and confidence. In this paper we describe Chunker's structure and performance, and discuss our plans for extending it to play the whole domain of king and pawn endings.

- [Carbonell 82] Carbonell, J.G.
Learning by Analogy: Formulating and Generalizing Plans from Past Experience.
Technical Report CMU-CS-82-126, Carnegie-Mellon University, Computer Science
Department,
June, 1982.

Analogical reasoning is a powerful mechanism for exploiting past experience in planning and problem solving. This paper outlines a theory of analogical problem solving based on an extension to means-end analysis. An analogical transformation process is developed to extract knowledge from past successful problem solving situations that bear strong similarity to the current problem. Then, the investigation focuses on exploiting and extending the analogical reasoning model to generate useful exemplary solutions to related problems from which more general plans can be induced and refined. Starting with a general analogical inference engine, problem solving experience is, in essence, compiled incrementally into effective procedures that solve various classes of problems in an increasingly reliable and direct manner.

- [Cole 83] Cole, R., R. Stern, S. Brill, M. Phillips, A. Pliant and P. Specker.
Feature-Based, Speaker-Independent, Isolated Letter Recognition.
In *ICASSP 83 Proceedings, IEEE ASSP*, 1983.

- [Cole 84] Cole, R.A., R.M. Stern, and M.J. Lasry.
Performing Fine Phonetic Distinctions: Templates vs. Features,
In J. Perkell, et al., *Invariance and Variability of Features in Spoken English Letters.*
Lawrence Erlbaum, New York, 1984.

This paper compares two basic approaches to computer speech recognition, template matching and feature-based recognition. In template matching, words are represented as spectral templates. During recognition, the input is compared to each stored template, time-frame by time-frame, and the best match is the recognized word. Feature-based systems extract phonetic features from the signal, such as formant frequencies and formant trajectories, and classify words in terms of an

expected set of feature values for each word. We show that template matching systems are unable to perform fine phonetic distinctions. A feature-based system is described which performs fine phonetic distinctions in a speaker-independent mode. It is shown that the success of the system depends critically on knowledge about the sources of variation in speech.

[Ebeling 84]

Ebeling, Carl, and Palay, Andrew.

The Design and Implementation of a VLSI Chess Move Generator.

In *Proceedings of the 11th Annual International Symposium on Computer Architecture*, IEEE Society, June, 1984.

Communication is a basic problem when using VLSI technology to implement large parallel circuits. Valuable chip area must be used to run wires connecting components on a chip and current packaging technology restricts the amount of communication that can cross chip boundaries. This paper presents a large parallel architecture for generating moves in chess and shows how it can be restructured to reduce communication and permit a straightforward VLSI implementation without any performance loss. The result is a move generator comprising 64 identical custom chips performing at a rate of 500,000 moves per second, performance that is comparable to the existing move generator.

The success of the architecture of a component module like a chess move generator depends not only on its performance but on how well it meshes with the rest of the system. We discuss the requirements of a chess move generator in the context of a chess-playing system and describe how each of these are met by our design. Details of the chip design are presented along with a description of how the move generator is built using identical chips.

[Fahlman 82]

Fahlman, S.F.

Three Flavors of Parallelism.

In *Proceedings of the Fourth National Conference, Canadian Society for Studies of Intelligence*, May, 1982.

This paper explores the relative costs and powers of three different kinds of parallel computing architecture that have been proposed for use in AI. Instead of parceling a problem out to a small, fixed number of processors, all of these systems employ a much higher degree of parallelism: they provide enough processing elements that we can assign a separate processor to every assertion in a large knowledge base, to every pixel in an image, or to every word in a speech system's lexicon. But, while these three kinds of system share this general orientation toward the massive use of parallelism, they differ markedly in the complexity of their processing elements and interconnections. They also differ in the kinds of problems that they can attack in parallel, without resorting to serial processing techniques. In order of increasing complexity and power, these categories are *marker-passing* systems, *value-passing* systems, and *message-passing* systems.

[Fahlman 83]

Fahlman, S.E., G.E. Hinton, and T.J. Sejnowski.

Massively Parallel Architectures for AI: NETL, Thistle, and Boltzmann Machines.

In *Proceedings of the AAAI-83 Conference*, AAAI, August, 1983.

August 22-26, 1983, in Washington, DC. Nominated, Publisher's Prize for Best Paper.

It is becoming increasingly apparent that some aspects of intelligent behavior require enormous computational power and that some sort of massively parallel computing architecture is the most plausible way to deliver such power. Parallelism, rather than raw speed of the computing elements, seems to be the way that the

brain gets such jobs done. But even if the need for massive parallelism is admitted, there is still the question of what kind of parallel architecture best fits the needs of various AI tasks.

In this paper we will attempt to isolate a number of basic computational tasks that an intelligent system must perform. We will describe several families of massively parallel computing architectures, and we will see which of these computational tasks can be handled by each of these families. In particular, we will describe a new architecture, which we call the Boltzmann machine, whose abilities appear to include a number of tasks that are inefficient or impossible on the other architectures.

[Forgy 81]

Forgy, C. L.

OPS5 User's Manual.

Technical Report CMU-CS-81-135, Carnegie-Mellon University, Computer Science Department,

July, 1981.

This is a combination introductory and reference manual for OPS5, a programming language for production systems. OPS5 is used primarily for applications in the areas of artificial intelligence, cognitive psychology, and expert systems. OPS5 interpreters have been implemented in LISP and BLISS.

[Forgy 84]

Forgy, C.L.

The OPS83 Report.

Technical Report CMU-CS-84-133, Carnegie-Mellon University, Computer Science Department,

May, 1984.

OPS83 is a programming language for expert systems applications. It combines the rule-based programming paradigm of the earlier versions of OPS with the procedural programming paradigm of conventional programming languages. It is less restrictive than the earlier versions of OPS in several respects, including the data structures permitted in working memory and the kinds of expressions that can be used in the LHSs of rules. OPS83 is a compiler-based language, and it provides for separate compilation of modules with full type checking across modules.

[Hood 82]

Hood, G. and J.G. Carbonell.

The World Modelers Project: Constructing a Simulated Environment to Aid AI Research.

In *Proceedings of the Thirteenth Annual Conference on Modeling and Simulation*, 1982.

The World Modelers Project is a physical simulation system designed to bridge the gap between real-world sensory and manipulator robotics, and Artificial Intelligence (AI) research on learning, problem solving, natural language processing and other cognitive phenomena presently investigated without benefit of direct interaction with the external world. In the simulation system, various 'organisms' controlled by AI programs or human users can perceive and act upon the simulated world, which in turn reflects changes caused by the (possibly concurrent) actions of organisms, according to internal laws of physics. This paper discusses the 3D simulation system, the graphics-based multi-window user interface, the distributed implementation, and implications for AI research.

[Kant 84]

Kant, E. and A. Newell.

Problem Solving Techniques for the Design of Algorithms.

Information Processing and Management 20(1-2):97-118, 1984.

Also available as CMU-CSD technical report CMU-CS-82-145.

By studying the problem-solving techniques that people use to design algorithms we can learn something about building systems that automatically derive algorithms or assist human designers. In this paper we present a model of algorithm design based on our analysis of the protocols of two subjects designing three convex hull algorithms. The subjects work mainly in a data-flow problem space in which the objects are representations of partially specified algorithms. A small number of general-purpose operators construct and modify the representations; these operators are adapted to the current problem state by means-ends analysis. The problems space also includes knowledge-rich schemas such as divide and conquer that subjects incorporate into their algorithms. A particularly versatile problem-solving method in this problem space is symbolic execution, which can be used to refine, verify, or explain components of an algorithm. The subjects also work in a task-domain space about geometry. The interplay between problem solving in the two spaces makes possible the process of discovery. We have observed that the time a subject takes to design an algorithm is proportional to the number of components in the algorithm's data-flow representation. Finally, the details of the problem spaces provide a model for building a robust automated system.

[Korf 83]

Korf, R. E.

Learning to Solve Problems by Searching for Macro-Operators.

Technical Report CMU-CS-83-138, Carnegie-Mellon University, Computer Science Department,

July, 1983.

PhD Thesis.

This thesis explores the idea of learning efficient strategies for solving problems by searching for *macro-operators*. A macro-operator, or *macro* for short, is simply a sequence of operators chosen from the primitive operators of a problem. The technique is particularly useful for problems with *non-serializable* subgoals, such as Rubik's Cube, for which other weak methods fail. Both a problem-solving program and a learning program are described in detail. The performance of these programs is analyzed in terms of the number of macros required to solve all problem instances, the length of the resulting solutions (expressed as the number of primitive moves), and the amount of time necessary to learn the macros. In addition, a theory of why the method works, and a characterization of the range of problems for which it is useful are presented. The theory introduces a new type of problem structure called *operator decomposability*. Finally, it is concluded that the macro technique is a valuable addition to the class of weak methods, that macro-operators constitute an interesting and important representation of knowledge, and that searching for macros may be a useful general learning paradigm.

[Laird 83]

Laird, J. and A. Newell.

A Universal Weak Method: Summary of Results.

In *Proceedings of the IJCAI-83*, Pages 771-773. International Joint Conference on Artificial Intelligence, Los Altos, CA, June, 1983.

Also appears as CMU-CSD technical report CMU-CS-83-141.

The weak methods occur pervasively in AI systems and may form the basic methods for

all intelligent systems. The purpose of this paper is to characterize the weak methods and to explain how and why they arise in intelligent systems. We propose an organization, called a *universal weak method*, that provides functionality of all the weak methods. A universal weak method is an organizational scheme for knowledge that produces the appropriate search behavior given the available task-domain knowledge. We present a problem solving architecture, called SOAR, in which we realize a universal weak method. We then demonstrate the universal weak method with a variety of weak methods on a set of tasks.

[Laird 84a]

Laird, John E., Rosenbloom, Paul S., and Newell, Allen.

Towards Chunking as a General Learning Mechanism.

In *Proceedings of AAAI-84 National Conference on Artificial Intelligence*, Pages 188-192.

American Association for Artificial Intelligence, 1984.

Chunks have long been proposed as a basic organizational unit for human memory.

More recently chunks have been used to model human learning on simple perceptual-motor skills. In this paper we describe recent progress in extending chunking to be a general learning mechanism by implementing it within a general problem solver. Using the Soar problem-solving architecture, we take significant steps towards a general problem solver that can learn about all aspects of its behavior. We demonstrate chunking in Soar on three tasks: the Eight Puzzle, Tic-Tac-Toe, and a part of the R1 computer-configuration task. Not only is improvement with practice, but chunking also produces significant transfer of learned behavior, and strategy acquisition.

[Laird 84b]

Laird, J. E.

Universal Subgoalting.

Technical Report CMU-CS-84-129, Carnegie-Mellon University, Computer Science

Department,

May, 1984.

The goal of this thesis is to develop a problem-solving architecture where all appropriate knowledge is brought to bear to control all aspects of problem-solving behavior. Such an architecture allows the creation of completely reflective problem solvers. We identify a specific capability called *universal subgoalting* that together with previous work on a *universal weak method* makes this possible. With universal subgoalting, subgoals arise whenever there is a difficulty in performing the problem-solving functions. In a subgoal, the problem solver brings its knowledge to bear to reason about and eliminate the difficulty. We identify a set of requirements that must be met by any problem-solving paradigm and architecture that realizes universal subgoalting. We then describe an implementation of universal subgoalting within *Soar*, a production system based on search in a problem space. We provide two demonstrations of universal subgoalting: (1) *Soar* creates subgoals whenever difficulties arise in any aspect of problem solving, (2) it is possible to encode the knowledge required to produce the weak methods so that the knowledge is used whenever it is needed. As part of the second demonstration, we provide a useful taxonomy of the weak methods based on the knowledge required to encode in *Soar*.

[Lasry 84]

Lasry, M.J. and R.M. Stern.

Unsupervised Adaptation to New Speakers in Feature-Based Letter Recognition.

In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Pages 17.6.1-17.6.3. IEEE Society, 1984.

This paper describes two new methods by which the CMU feature-based recognition system can learn the acoustical characteristics of individual speakers without feedback from the user. We have previously described how the system uses MAP techniques to update its estimates of the mean values of features used by the classifier in recognizing the letters of the English alphabet on the basis of a priori information and labeled observations. In the first of the new procedures described in this paper the system assumes a correct decision every time it classifies a new utterance with a sufficiently high confidence level. In the second new procedure the system adjusts its estimates of the means on the basis of their correlation with the average values of the features over all utterances. In each case classification performance using the unsupervised estimation procedures could equal that obtained using speaker adaptation with feedback from the user, although which method provided the better performance depended on which set of letters was being classified.

[McDermott 82] McDermott, J.

R1: A Rule-based Configurer of Computer Systems.

Artificial Intelligence 19(1):39-88, September, 1982.

R1 is a program that configures VAX-11/780 computer systems. Given a customer's order, it determines what, if any, modifications have to be made to the order for reasons of system functionality and produces a number of diagrams showing how the various components of the order are to be associated. The program is currently being used on a regular basis by Digital Equipment Corporation's manufacturing organization. R1 is implemented as a production system. It uses Match as its principal problem solving method: it has sufficient knowledge of the configuration domain and of the peculiarities of the various configuration constraints that at each step in the configuration process, it simply recognizes what to do. Consequently, little search is required in order for it to configure a computer system.

[McKeown 83] McKeown, D.M., Jr., and J. McDermott.

Toward Expert Systems for Photo Interpretation.

In *IEEE Trends and Applications* 1983, Pages 33-39. May, 1983.

Available as reprint no. CH1887-9/83/0000/0033 from the IEEE Society, P.O. Box 80452, Worldway Postal Center, Los Angeles, Ca. 90080.

In this paper we describe some preliminary results in the design and implementation of a system for semi-automatic photo-interpretation of high resolution aerial photographs. The system, SPAM, consists of three major components, an image/map database, a collection of image processing tools, and a rule-based system whose domain of expertise is commercial airports in general, and the National Airport (Washington D.C.) in particular. We present our design rationale, describe those components which have been implemented, and discuss design and implementation currently in progress. Applications for such photo-interpretation systems include cartography and decision support systems for situation assessment.

[McKeown 84] McKeown, D.M., W.A. Harvey, and J. McDermott.

Rule Based Interpretation of Aerial Imagery.

In *Proceedings of the 1984 IEEE Workshop on Principles of Knowledge-Based Systems*, 1984.

In this paper we describe the organization of a rule based system, SPAM, that uses map and domain specific knowledge to interpret airport scenes. This research investigates the use of rule-based system for the control of image processing and interpretation of results with a respect to a world model, as well as the representation of

the world model within an image/map database. We present results on the interpretation of an high resolution airport scene where the image segmentation has been performed by a human and by a region-based image segmentation program. The result of the system's analysis is characterized by the labeling of individual regions in the image and the collection of these regions into consistent interpretations of the major components of an airport model. These interpretations are ranked on the basis of their overall spatial and structural consistency. Some evaluations based on the results from three evolutionary versions of SPAM are presented.

[Newell 80]

Newell, A.

Reasoning, Problem Solving, and Decision Processes: The Problem Space as a Fundamental Category.

Lawrence Erlbaum Associates, Hillsdale, NJ., 1980.

See Chapter 35.

The notion of a problem space is well known in the area of problem solving research, both in cognitive psychology and artificial intelligence. The *Problem Space Hypothesis* is enunciated that the scope of problem spaces is to be extended to all symbolic cognitive activity. The chapter is devoted to explaining the nature of this hypothesis and describing some of its potential implications, with no attempt at a critical marshalling of the evidence pro and con. Two examples are used, one a typical problem solving activity (the Tower of Hanoi) and the other syllogistic reasoning. The latter is an example where the search behavior typical of problem spaces is not clearly in evidence, so it provides a useful area to explore the extension of the concept. A focal issue used in the chapter is the origin of the numerous flow diagrams that serve as theories of how subjects behave in tasks in the psychological laboratory. On the Problem Space Hypothesis these flow diagrams derive from the interaction of the task environment and the problem space.

[Palay 83]

Palay, A.J.

Searching with Probabilities.

Technical Report CMU-CS-83-145, Carnegie-Mellon University, Computer Science Department,

July, 1983.

In this thesis we investigate two issues relating to heuristic search algorithms. The first and most important issue addressed is the technique used to represent knowledge within a search tree. Previous techniques have used either single values or ranges. We demonstrate that probability distributions, using a modified B*-type search algorithm, can be used successfully as a knowledge representation technique: Our experiments show that the probability-based algorithm is able to solve a wide variety of tactical chess problems. Furthermore, using both analytical examples and experimental results, we show that the use of probability distributions is superior to the use of either of the previous techniques. Experimentally we show that the probability-based algorithm solves over one-third more problems than the comparable range-based algorithm and expands approximately one-tenth the nodes on problems that both algorithms solve. We also show, again within the domain of tactical chess problems, that the probability-based algorithm is better than any alpha-beta program that searches to an average depth of six-ply or less. The second issue addressed in this thesis is the development of a method that can be used to generate range-based (and probability-based) knowledge representations. The inability to generate reasonable ranges has been a major obstacle to

testing the B* algorithm. We present one method based on the use of a null-move search that can be used for generating ranges (and distributions) within the domain of chess.

[Rosenbloom 81] Rosenbloom, Paul S.

A World-Championship-Level Othello Program.

Technical Report CMU-CS-81-137, Carnegie-Mellon University, Computer Science Department,

August, 1981.

Othello is a recent addition to the collection of games that have been examined within artificial intelligence. Advances have been rapid, yielding programs that have reached the level of world-championship play. This article describes the current champion Othello program, *Iago*. The work described here includes: (1) a task analysis of Othello; (2) the implementation of a program based on this analysis and state-of-the-art AI game-playing techniques; and (3) an evaluation of the program's performance through games played against other programs and comparisons with expert human play.

[Rosenbloom 82] Rosenbloom, P.S. and A. Newell.

Learning by Chunking: Summary of a Task and a Model.

In *Proceedings of the AAAI-82 National Conference on Artificial Intelligence*, Pages 255-257.

The American Association for Artificial Intelligence, August, 1982.

The *power law of practice* states that performance on a task improves as a power law function of the number of times the task has been performed. In this article we describe recent work on a model of this effect. The model, called the *chunking theory of learning*, is based on the notion of *chunking*. A limited version of this model has been implemented within the Xaps2 production system architecture. When it is applied to a 1023-choice reaction-time task (encoded as a set of productions), task performance is improved (measured in terms of the number of production system cycles). Moreover, the practice curves are power law in form.

[Rosenbloom 83] Rosenbloom, P.S.

The Chunking of Goal Hierarchies (A Model of Practice and Stimulus-Response Compatibility).

Technical Report CMU-CS-83-148, Carnegie-Mellon University, Computer Science Department,

August, 1983.

In this thesis we present an integrated theory for two phenomena: *practice* and *stimulus-response compatibility*. It is a theory that is both psychologically plausible and useful as a learning mechanism for AI systems.

The work on practice is based on our earlier investigations that showed that: (1) when human performance is measured in terms of the time required to do a task, it improves as a power-law function of the number of times the task has been performed; and (2) that a model of practice based on the concept of *chunking* was capable of producing power-law practice curves.

The previous work established the feasibility of the chunking theory for a single task, but the implementation was specific to that one task. In this thesis we develop a modified formulation of the chunking theory that allows a more general implementation. In this formulation, task algorithms are expressed in terms of hierarchical goal structures. These algorithms are simulated within a goal-based production-system architecture, designed for this purpose. It improves the perfor-

mance of the system by gradually reducing the need to decompose goals into their subgoals.

This model has been successfully implemented and applied to the task employed in the previous work, and to a set of variations on three stimulus-response compatibility tasks. Compatibility is a topic for which there is still no metric theory. We provide two formulations of such a theory. Both formulations of this model provide good fits to the data from the three compatibility experiments.

[Rosenbloom 84] Rosenbloom, P.S., Laird, J.E., McDermott, J., Newell, A., and E. Orciuch.

R1-Soar: An Experiment in Knowledge-Intensive Programming in a Problem-Solving Architecture.

Pattern Analysis and Machine Intelligence 7:561-569, January, 1984.

Also available in CMU-CSD technical report CMU-CS-85-110, and the *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, IEEE Computer Society, December 1984.

This paper presents an experiment in knowledge-intensive programming in Soar. In Soar, knowledge is encoded within a set of problem spaces, yielding a system capable of reasoning from first principles. Expertise consists of additional rules that guide complex problem-space searches and substitute for expensive problem-space operators. The resulting system uses both knowledge and search when relevant. Expertise knowledge is acquired either by having it programmed, or by a chunking mechanism that automatically learns new rules reflecting the results implicit in the knowledge of the problem spaces. The approach is demonstrated on the computer-system configuration task, the task performed by the expert system, R1.

[Rudnick 82] Rudnick, A.I., A.H. Waibel, and N. Krishnan.

Adding a Zero-Crossing Count to Spectral Information in Template-Based Speech Recognition.

Technical Report CMU-CS-82-140, Carnegie-Mellon University, Computer Science Department, October, 1982.

Zero-crossing data can provide important feature information about an utterance which is not available in a purely spectral representation. This report describes the incorporation of zero-crossing information into the spectral representation used in a template-matching system (CICADA). An analysis of zero-crossing data for an extensive (2880 utterance, 8 talker) alpha-digit data base is described. On the basis of this analysis, a zero-crossing algorithm is proposed. The algorithm was evaluated using a confusable subset of the alpha-digit vocabulary (the 'E-set'). Inclusion of zero-crossing information in the representation leads to a 10-13% reduction in error rate, depending on the spectral representation.

[Stern 83] Stern, R.M. and M.J. Lasry.

Dynamic Speaker Adaptation for Isolated Letter Recognition Using MAP Estimation.

In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Pages 734-737. IEEE, April, 1983.

A dynamic speaker-adaptation algorithm for the CMU feature-based isolated letter recognition system, FEATURE, is described. The algorithm, based on maximum a posteriori probability estimation techniques, uses the labeled observations input thus far to the classifier, as well as the *a priori* correlations of the features within and across the various letters of sets of letters (classes). The probability density

functions (pdf) of all the classes are updated simultaneously rather than on a class-by-class basis so that the pdf of a given class is updated before any observation from that class has been input. A significant improvement in the recognition performance was observed for different vocabularies as the system tuned to the characteristics of a new speaker. Finally, the algorithm was compared to simpler forms of dynamic adaptation. It produced a faster decrease of the error rate than the other tuning procedures. After a small number of iterations, however, the various procedures yielded similar results.

- [Touretzky 84] Touretzky, D.S.
The mathematics of inheritance systems.
 PhD thesis, Carnegie-Mellon University, Computer Science Department, May, 1984.

- [Waibel 81] Waibel, A. and B. Yegnanarayana.
Comparative Study of Nonlinear Time Warping Techniques in Isolated Word Speech Recognition Systems.
 Technical Report CMU-CS-81-125, Carnegie-Mellon University, Computer Science Department,
 June, 1981.

In this paper we present the description of an isolated word recognition system and a discussion of various design choices that affect its performance. In particular, we report experimental results aimed at evaluating several methods to optimize the performance of dynamic warping algorithms. Three major aspects that have been suggested in the literature have been investigated: (1) relaxation of the boundary conditions to allow for inaccurate begin-end time detection, (2) choice of warping algorithm, e.g., Itakura asymmetric, Sakoe and Chiba symmetric, Sakoe and Chiba asymmetric, and (3) choice of an appropriate warping window to restrict computation to a minimum needed for best recognition results. Recognition results were tested on two vocabularies: the digits and a highly confusable subset of the alphabet (e.g., e,b,d,p,t,g,v,c,z). (1) The relaxation of the boundary conditions degraded the performance of the confusable subset and the digits. (2) The asymmetric Itakura algorithm yielded better results for the confusables, while we obtained slightly better results for the digits using the symmetric Sakoe and Chiba algorithm. (3) The choice of a 100-ms warping window appears to be optimal for both vocabularies used.

- [Waibel 82] Waibel, A.
Towards Very Large Vocabulary Word Recognition.
 Technical Report CMU-CS-82-144, Carnegie-Mellon University, Computer Science Department,
 November, 1982.

In this paper, preliminary considerations and some experimental results are presented in an effort to design Very Large Vocabulary Recognition (VLVR) systems. We will first consider the applicability of current recognition techniques and argue their inadequacy for VLVR. Possible alternate strategies will be explored and their potential usefulness statistically evaluated. Our results indicate that suprasegmental cues such as syllabification, stress patterns, rhythmic patterns and the voiced-unvoiced patterns in the syllables of a word provide powerful mechanisms for search space reduction. Suprasegmental features could thus operate in a complementary fashion to segmental features.

5. Research in Programming Technology

Programming Technology concerns all aspects of constructing high-quality hardware/software systems. By "high-quality", we mean systems that are: demonstrably correct, resource-efficient, produced on time and within budget, and easily maintained and enhanced. Moreover, such systems must provide a flexible environment for diverse users. Research in programming technology addresses the principles, knowledge, and tools (compilers, debuggers, editors, design systems, etc.) used to produce software systems. At CMU, in particular, our goal is to enhance the ability to produce *predictably* high-quality hardware/software systems. We measure progress toward that goal either in terms of increased quality of particular systems (e.g. compilers) or greater complexity in tasks we can produce at a given quality.

Through our research we have developed techniques, languages, and methodologies that improve programming productivity and produce high-quality systems. The three key areas we focussed on were:

- Automated compiler construction
- Highly secure and reliable systems
- Advanced programming environments

5.1. Automating Compiler Construction

Productivity increases when programmers can work in high-level languages that effectively capture human reasoning. The compilers and interpreter that translate such languages into hardware-specific machine code represent some of our most important programming tools. Though we have been writing them for almost three decades, they remain large and complex. Writing a compiler by hand, even with all our experience, typically requires ten to twelve man years. Exacerbating the problem is the fact that each unique combination of language, operating system, peripheral equipment, hardware, etc. can require a new compiler. Our research has shown how to automate the job by using generic compiler-generating systems. Current techniques reduce the creative effort to approximately one man-year and promise significant improvements in developing more general programming systems. Automating compiler construction has three major aspects, discussed in the following sections:

- Analyzing the target language
- Developing effective "generating-system" techniques
- Discovering code-optimizing compiler techniques

5.1.1. Analyzing the target language

Commercial firms are currently testing compiler-generator systems for traditional Algol-like languages (e.g. C, Pascal, Modula-2). Meanwhile, CMU researchers are experimenting and refining systems for , an advanced Algol-like language. Our strategy is to understand the challenge of writing Ada compilers by examin-

ing the language's strengths and weaknesses and experimenting with compiler techniques that support the specific features of Ada not found in other imperative languages such as C or Pascal.

We have evaluated Ada and compared it with other programming languages such as Pascal [Habermann 81a, Hibbard 83, Shaw 81, Shaw 84a]. Our work has demonstrated the fundamental use of data abstraction in Ada has revealed Ada's flexibility and expressive richness [Perlis 81, Shaw 82]. While rich at the source level, Ada also permits efficient machine code, and a smart compiler can "compile out" the complexity to achieve efficiency comparable to what Pascal and C compilers produce. Designing such clever Ada compilers has turned out to be far more difficult than we anticipated in 1981.

CMU researchers have developed runtime support techniques and various code generation and optimization methods especially for Ada [Bentley 81a]. We designed intermediate representations of Ada programs in IDL (Interface Description Language) [Nestor 81] and Diana, which has become an international standard [Goos 81]. These representations describe the data used to communicate among collections of related programs, such as the set of tools in a programming environment [Garlan 84, Barbacci 82a]. Current compilers can generate code correctly with acceptable efficiency. We expect optimizing Ada compilers to reach the market by the end of the decade.

5.1.2. Creating "generating-systems" techniques

A compiler consists typically of three parts. The front end checks the syntax and semantics of the input program; the intermediate processor transforms the control flow output of the front end into pseudo machine code; the back end transforms the pseudocode into genuine machine code. The technique of creating systems that generate software products can be applied to the various stages of a compiler and to the production of software systems in general. We asked the following questions:

- Which part of the production process can be automated?
- How should we integrate generated parts with existing code?
- How can we insert special system requirements?
- How should the system represent and process target system descriptions?

Generators have been produced for the front end of compilers, particularly for handling syntax. Creating generators for the back end is challenging because no description formalism exists as for front end (BNF), and quality code generators require sophisticated code optimization specific to the target system.

Our main achievement in this area is the design and implementation of a Back End Compiler Generator System: PQCC (Production Quality Compiler Compiler). PQCC is a sophisticated system that produces efficient code generators for various machine architectures. PQCC requires three types of input:

- an IDL description in intermediate code

- an ISPS description of the underlying hardware
- a macro library of hardware-specific code optimizations

PQCC has mastered Modula-2, the intermediate step from Pascal to Ada. The system has been perfected to the point where PQCC is superior to commercial products for Pascal and C, and can handle machines of the DEC and IBM variety.

5.1.3. Finding code optimizing compiling techniques

The final aspect of a Compiler Generator that we investigated was compiling techniques with code optimization. As mentioned earlier, quality code generators require sophisticated code optimization. To achieve the required level of code optimization, we considered the representation of the intermediate code, the input describing the machine hardware, and known optimization techniques. CMU produced two demonstration systems of Ada environments. The main focus of the Gandalf project, the first system, is automation and generic programs(see Section 5.3) in generating an Ada environment [Habermann 81b]. The Gandalf-generated Ada environment's knowledge of Ada syntax and semantics prevents the user from making syntactic and semantic errors. An Ada program generated by Gandalf is guaranteed to be free of syntax errors and static semantic violations.

CMU's second demonstration system of an Ada environment, the Ada+ project, focussed on integrating an Ada compiler with an operating system and extending the operating system with Ada tools. The project has produced an Ada compiler on the Spice-Accent operating system, concentrating on version control tools and automatic recompilation [Habermann 81b]. As a side effect of this project, we have gathered useful information on peculiarities of Ada code generation.

5.2. Highly Secure and Reliable Systems

In a secure, reliable system, the key programs must be absolutely correct and circumventing them at runtime should be extremely difficult. The requirement for absolute correctness puts a tremendous burden on system designers to show that their design is secure and on implementors to demonstrate that their product accurately reflects the design. Achieving runtime reliability demands that designers compensate for inevitable hardware failures. Our key strategies for meeting the dual challenge of security and reliability are redundancy and verification.

Redundancy permits reliability checks and protects against hardware failures but has been unpopular in uniprocessor machines because it adds execution time overhead. However, multiprocessing systems, distributed systems, and computer networks, through their parallelism and concurrency, now offer promising

solutions to the time problem. Decreasing hardware costs, of both processors and memory, make it increasingly feasible to trade processing cycles for redundant hardware.

The programming technology community has mastered verification for sequential programs in standard programming languages, but large systems and parallel or concurrent computations pose additional problems. Techniques suitable for sequential programs fail to capture concurrency concepts such as temporal event ordering. Fortunately, research on verification thrives in computational models abstracted from real system designs. One can, for example, investigate how to synchronize or schedule events in distinct processes without considering the specific tasks these processes carry out.

Our research has stressed relevance to working systems. We have pursued three parallel paths that combine reliability and verification goals within real multiprocessor environments:

- Verification concepts, techniques, and applications on parallel architectures
- Design and implementation of parallel architectures and their software
- Fault tolerance and reliability support in parallel systems.

5.2.1. Design and implementation of parallel architectures and software

Research in parallel architectures for security and reliability has led to the design and implementation of both hardware and software for tightly coupled multiprocessors, loosely coupled multiprocessors, and systems distributed on computer networks.

CMU's multiprocessor research spans the past 15 years. Our design and implementation of tightly coupled multiprocessors goes back to the C.mmp machine which consisted of sixteen processors having access to any one of sixteen 256K memory units on an instruction by instruction basis. We built a loosely coupled multiprocessor, Cm*, out of variable clusters of processors and memories connected through computer nodes responsible for translating remote program and data calls(Kmaps). This architecture is particularly suited for increasing reliability through redundancy [Harbison 82a]. The Kmap translation step facilitates the implementation of security because it includes both address translation and authorization checks. Applications on this architecture show great promise for speeding up execution time.

Over the past five years, CMU has gained extensive experience in networking and using distributed systems on networks. Reliability can be greatly enhanced by replication and redundancy. This has been demonstrated in the Distributed Sensor Net project (see Chapter 6) of which the Spice project is an offspring [Harbison 82b]. A particular issue of protection arises if simultaneous access to resources is a potential threat to consistency of the network state. Our work has shown the general applicability of the concept of "atomic transactions" and how these can be implemented on local area networks for a variety of tasks.

5.2.2. Verification concepts, techniques, and applications on parallel architectures

Testing serves as an engineering aid to observe behavior and fine tune systems. While testing can show the presence of flaws or errors, but not their absence, verification assures the absolute correctness of an implementation.

One of the most productive techniques for sequential programs is Hoare's axiomatic approach. This approach translates each language construct into a statement in elementary logic: "A and B", "A implies B", or "not A." Extending this to concurrent computations, "temporal logic" allows statements involving time, such as "at some time in the future" or "will be true forever" [Clarke 83a]. We designed and implemented a temporal logic verification system. It can handle parallel designs that can be translated into a finite state transition diagram. Network protocols and process communication on parallel architectures satisfy this condition. The only systems excluded from this verification method are those that can add or generate resources dynamically [Clarke 82a, Clarke 83b]. A particular achievement has been the use of the verification system for proving the correctness of some VLSI chip designs. These designs are typically described by a diagram of more than 1000 states. In several cases, this verification system found subtle design flaws that had escaped the most careful scrutiny of its designers and testers [Clarke 83c]. Researchers are extending these verification techniques to apply to transactions in concurrent computations [Spector 83].

5.2.3. Specific design of fault tolerance and reliability support in parallel systems

Although hardware failure is inevitable, it can be counteracted by building fault-tolerant systems that make heavy use of redundancy and replication. Our research has resulted in the design and analysis of a variety of reconfiguration strategies under the assumption that parts of a system malfunction [Clarke 82b, Harbison 82b]. An important experimental system under construction for increasing reliability is TABS, Transaction-Based Distributed Systems. TABS achieves reliability mainly through atomic transactions that guarantee either complete failure of a response to a request or complete success. The network will never get into an intermediate state in which a modification has partially succeeded, but failed somewhere in the middle [Spector 83].

Reliability requirements are not uniform for different applications and/or architectures. Therefore, it is necessary to provide user facilities to express the degree of reliability required in their programming language [Durham 82]. We demonstrated the usefulness of the parallel approach to redundancy with performance studies of various architectures [Nestor 81].

5.3. Advanced Programming Environments

Programming environment research is developing systems that will support both programming-in-the-large (assembling program modules into complete software systems) and programming-in-the-small (writing the individual component programs). Such environments are crucial for resolving the serious inability to produce and maintain large, reliable software systems. In the past, tools (e.g. editors, compilers, etc.) have been designed and implemented to support particular aspects of the system design process. Despite their valuable contributions, these tools suffer two serious drawbacks: typically designed in isolation, each having its own idiosyncratic interface, they show little regard for other tools; they are handcrafted and are therefore difficult to modify or adapt.

Ultimately, we want to build programming environments that serve as intelligent assistants and not merely as tool kits. Thus our goal is to provide a collection of integrated, interactive tools that users can easily modify or extend to meet changing needs at both system and module levels. To reach that goal by the end of this decade, our work explores the merging of programming environment research from the last five years with advances in knowledge-based, expert systems.

5.3.1. Designing a programming environment

Our target programming environment is one that supports system developers by:

- Assisting in writing individual programs
- Maintaining system version control
- Coordinating project management
- Providing a uniform user interface
- Integrating tools through a common database

To develop this intelligent environment, we coordinated common facilities, language syntax and semantics, and developer expertise. We identified and implemented facilities common to a class of programming environments (e.g. memory and management of input/output and windows). Researchers integrated the various tools by developing a common database that provides support for meeting syntax and semantics requirements, and then incorporated expert techniques for using the database. In order to implement our generation scheme, we had to integrate the implemented parts with an existing operating system. We chose UNIX to achieve portability. We also devised descriptive tools to allow designers to define the specific facilities of a programming environment.

5.3.2. Automatically generating environments

To most easily obtain extensible and modifiable tools, we concentrated on techniques that can generate programming environments automatically. Our basic strategy exploits ideas from automated compiler construction. Figure 5-1 depicts the scheme for generating a programming environment.

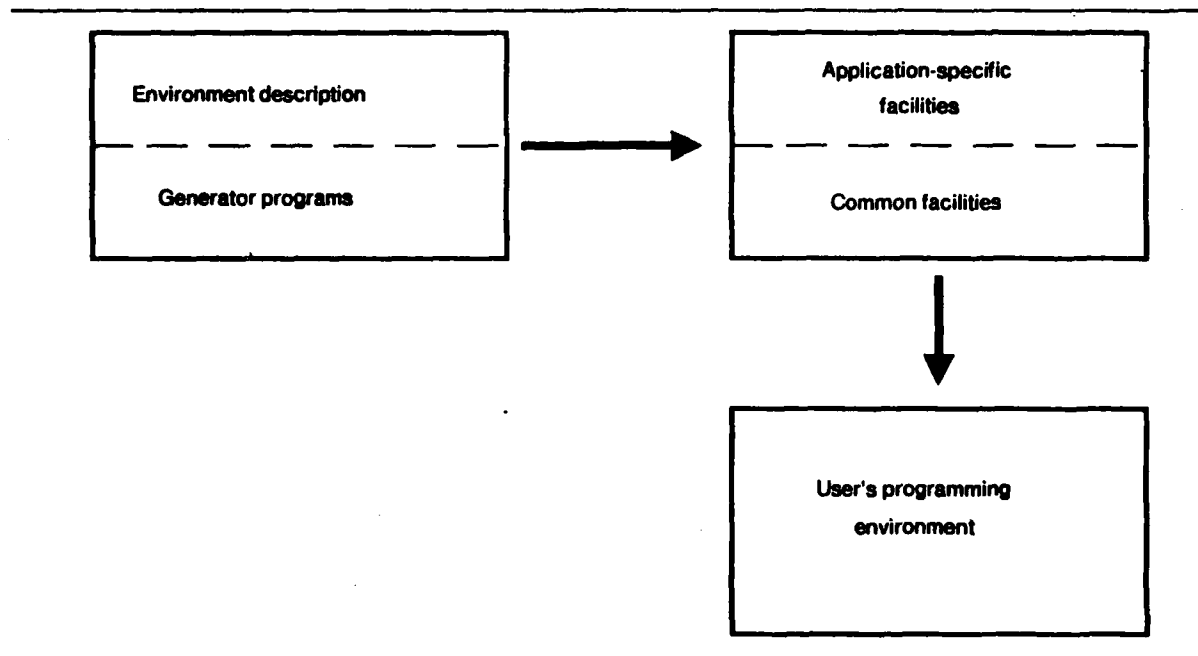


Figure 5-1: Generating a programming environment

Our ability to design and implement the Gandalf system demonstrates significant progress in advanced programming environments. The Gandalf System is a support environment for both the programming environment designer and the tool builder. In addition to our work with Gandalf, we emphasized providing supporting environments for the common facilities design [Ball 81, Ball 82]. We designed common facilities for memory management, database management, and for the user interface, particularly for workstations and terminals.

The Gandalf system runs on UNIX and provides specific support for describing the syntax and semantics of specific tools [Habermann 81b, Habermann 81c]. The most recent additions to the system are facilities for describing runtime support in programming environments [Leverett 82, Garlan 84]. Gandalf implements runtime support in programming environments through "active database" concepts borrowed from AI. Objects contain not only passive data, but also active programs (daemons) that function as watchdogs. The daemon facility is general enough to implement runtime semantic checking and various management functions, including access authentication and project management [Garlan 84, Habermann 81c]. Gandalf has been used for creating several programming environments for a variety of purposes. Two environments

deserve particular mention:

- The Gandalf C prototype
- The Gnome System

The Gandalf C prototype integrates programming-in-the-small with programming-in-the-large by providing a uniform user interface that defines all interactions in terms of editing operations. The compiler and debugger are integrated into the environment and incremental compilation is automatically performed when possible. Debugging takes the form of editing a running program [Habermann 81b]. Programming-in-the-large focuses on system configuration control. The user can define modular interfaces, versions and system configurations that group these versions into executable systems [Goos 81, Habermann 82a, Habermann 82b].

The purpose of the Gnome System is to support novice programmers learning to program in Pascal. The system provides incremental compilation and has complete knowledge of Pascal's syntax and semantics. Gnome is an interactive system (as all Gandalf products are) that constructs the programs in response to user commands. Since all programs are written by the system, syntax and semantics errors cannot arise. The system provides all the advantages to its users of interacting with an intelligent assistant who knows the rules perfectly. This allows the user to concentrate on substance rather than on form. The Gnome System is a high quality product that has gone through several design and implementation iterations [Feiler 82].

5.4. Bibliography

- [Ambriola 84] Ambriola, V., R.J. Ellison and G. Kaiser.
An Action Routine Model for ALOE.
 Technical Report CMU-CS-84-156, Carnegie Mellon University Computer Science Department,
 August, 1984.

We propose a new model for the design and implementation of structure editing environments for programming-in-the-small. Unlike previously proposed models, our model is an attempt at a complete solution that encompasses syntax, static semantics, and dynamic semantics. The essence of the model is that the pure syntax-directed editor is viewed as an abstract machine for the particular language. The semantics of the corresponding editing environment are implemented in terms of the tree-oriented machine language defined by the abstract machine. The semantics may be written using these primitives directly, in a higher-level tree-oriented programming language, or in a very high-level declarative notation that can be translated into one of the two types of executable language. In this document, we present the model itself and introduce a tree-oriented programming language for writing semantics in the framework of this model. To make the discussion more concrete, we explain the model in the context of the ALOE editor generator.

- [Atlas 83] Atlas, B. and Z. Segall.
Behavioral Analysis of an ISPS Oriented Architecture.
Computer Languages and Their Applications, May, 1983.

- [Ball 81] Ball, J.E.
 Canvas: The Graphics Package for the Spice Personal Timesharing System.
 In *Proceedings of Computer Graphics 81*, October, 1981.

- [Ball 82] Ball, J.E. and P.J. Hayes.
 A Test-bed for User Interface Designs.
 In *Proceedings of the Conference on Human Factors in Computer Systems*, March, 1982.

There is currently much well founded concern with interactive computer interfaces from the human factors point of view. Typically, interactive command interfaces appear to their users as unfriendly, uncooperative, and inflexible inhibitors of access to the underlying tool systems, rather than the facilitators they should be.

Over the past few years, we have been working towards the construction of improved user interfaces. In particular, we have been interested in interactive command interfaces that appear to their users much more friendly and cooperative than those presently available. During the course of our work, we have identified several features, not typically found in current interfaces, that we believe essential for cooperative interface behavior. These features include: resistance to trivial errors, ready availability of appropriate help text, continuity mechanism, personalization, and consistency across subsystem boundaries.

An attractive approach to measuring the effectiveness of some of these more advanced interface features, and thus either justifying them or showing them to be ineffective or not worth their price, is to build a test-bed interface in which the features can be implemented and evaluated through use. We are currently engaged in the construction of such a test-bed interface. We list a set of design goals or principles that we have followed: maximum separation between tools and interface, ef-

iciency, built-in instrumentation, and operation on widely available hardware.

- [Barbacci 82a] Barbacci, M.R.
Intermediate Representation for the Spice Ada+ Compiler.
Spice Report 138 , Carnegie Mellon University Computer Science Department,
September, 1982.
- [Barbacci 82b] Barbacci, M.R.
The Ada+ Programming Environment.
Spice Report 139 , Carnegie Mellon University Computer Science Department,
September, 1982.
- [Bentley 81a] Bentley, J.L.
Writing Efficient Code.
Technical Report CMU-CS-81-116, Carnegie Mellon University Computer Science Department,
April, 1981.
The most important step in making software systems efficient is the proper selection of data structure and algorithms; many papers and textbooks have been devoted to these topics. Most discussions, however, neglect another important activity: that of writing machine independent efficient code. This paper examines a set of techniques for accomplishing that step. We will examine those techniques both in an abstract setting and in their application to a real program, where they led to a speedup of a factor of over six. Because these techniques should be employed rarely, an important part of this paper is describing exactly when one should (and should not!) use them.
- [Bentley 81b] Bentley, J. L., D. F. Stanat, and J. M. Steele.
Analysis of a Randomized Data Structure for Representing Ordered Sets.
In *Proceedings of the Nineteenth Annual Allerton Conference on Communication, Control, and Computing*, October, 1981.
- [Bentley 81c] Bentley, J. L. and T. Ottmann.
The Complexity of Manipulating Hierarchically Defined Sets of Rectangles.
In *Proceedings of the Tenth International Symposium on the Mathematical Foundations of Computer Science*, August, 1981.
- [Bentley 81d] Bentley, J. L.
Squeezing Constant Factors of Geometric Algorithms.
In *Proceedings of the Nineteenth Annual Allerton Conference on Communication, Control, and Computing*, October, 1981.
- [Bentley 82] Bentley, J. L.
Writing Efficient Programs.
Prentice-Hall, 1982.
- [Bentley 83a] Bentley, J.L.
A Case Study in Writing Efficient Programs.
Technical Report CMU-CS-83-108, Carnegie Mellon University Computer Science Department,
January, 1983.
The time and space performances of a computer program rarely matter, but when they

do they can be of crucial importance to the success of the overall system. This paper discusses the performance issues that are aroused in the implementation of a small system (twelve programs, two thousand lines of code) for a small business. The paper emphasizes a general methodology for improving system performance; the details of the case study show how the general techniques are applied in a particular context.

- [Bentley 83b] Bentley, J.L. and C.C. McGoech.
Worst-Case Analyses of Self-Organizing Sequential Search Heuristics.
 Technical Report CMU-CS-83-121, Carnegie Mellon University Computer Science Department,
 March, 1983.

The performance of sequential search can be enhanced by the use of heuristics that move elements closer to the front of the list as they are found. Previous analyses have characterized the performance of such heuristics probabilistically. In this paper we show that the heuristics can also be analyzed in the worst-case sense, and that the relative merit of the heuristics under this analysis is different than in the probabilistic analyses. Simulations show that the relative merit of the heuristics on real data is closer to that of the new worst-case analyses rather than that of the previous probabilistic analyses.

- [Brookes 83a] Brookes, S.D., and W.C. Rounds.
Behavioral Equivalence Relations Induced by Programming Logics.
 Technical Report CMU-CS-83-112, Carnegie Mellon University Computer Science Department,
 March, 1983.

In this paper we compare the descriptive power of three programming logics by studying the elementary equivalence relations which the logics induce on nondeterministic state-transition systems. In addition, we compare these relations with other natural state-equivalence relations for nondeterministic systems. We find that the notions of *bisimilarity* and *observation equivalence* are very strong equivalences compared with those induced by logics. These three comprise *regular trace logic* (RTL), *propositional dynamic logic* (PDL), and *Hennessy-Milner logic* (HML). Regular trace logic is a new logic which can be used to give behavioral specifications for concurrent systems. It is a way of formalizing those properties of programs which have been given informally in terms of path expressions. The model theory and axiomatics of this logic are interesting in their own right. Propositional dynamic logic is well-known; our treatment differs from the standard one only in that we regard the modalities as specifying intended behavior instead of being programs. Hennessy-Milner logic is a simplified modal logic which those authors used as a characterization of their notion of observation equivalence, which we call weak observation equivalence in this paper. We also include a brief treatment in this context of two other natural equivalences for nondeterministic systems: *failure equivalence* and *trace equivalence*, both of which are weaker than the relations induced by the logics but can be characterized using appropriate logical subsets.

- [Brookes 83b] Brookes, S.D.
 On the Relationship of CSS and CSP.
 In *Proceedings of ICALP-83, Springer-Verlag Publishers*, July, 1983.
 This paper compares two models of concurrency, Milner's Calculus of Communicating

Systems (CCS) and the *failures* model of Communicating Sequential Processes (CSP) developed by Hoare, Brookes and Roscoe. By adapting Milner's synchronization trees to serve as notation for both CCS and CSP, we are able to define a representation mapping for CSP processes. We define an equivalence relation on synchronization trees which corresponds precisely to the notion of *failure equivalence*. This equivalence relation identifies two trees if and only if the processes represented by the trees have identical failure sets.

- [Brookes 83c] Brookes, S.D.
A Semantics and Proof System for Communicating Processes.
In *Proceedings of the 1983 Workshop on Logic of Programs* Springer-Verlag, June, 1983.
- [Brookes 84] Brookes, S.D., C.A.R. Hoare, and A.W. Roscoe.
A Theory of Communicating Sequential Processes.
Journal of the ACM, July, 1984.
- [Bruegge 84] Bruegge, B. and P.G. Hibbard.
Generalized Path Expressions: A High-Level Debugging Mechanism.
Journal of Systems and Software 3:265-276, 1984.
- [Chazelle 82a] Chazelle, B.
The Bottom-Left Bin-Packing Heuristic: An Efficient Implementation.
Technical Report CMU-CS-82-104a, Carnegie Mellon University Computer Science Department,
May, 1982.
- [Chazelle 82b] Chazelle, B.
An Improved Algorithm for the Fixed-Radius Neighbor Problem.
Technical Report CMU-CS-82-109, Carnegie Mellon University Computer Science Department,
May, 1982.
- [Chazelle 82c] Chazelle, B.
The Polygon Containment Problem.
Technical Report CMU-CS-82-106, Carnegie Mellon University Computer Science Department,
May, 1982.
- [Clarke 82a] Clarke, E.M., A.P. Sistla, N. Francez, and Y. Gurevich.
Can Message Buffers be Characterized in Linear Temporal Logic?
In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, August, 1982.
Also appears in *Information and Control*, Vol. 63 (1/2), Oct/Nov., 1984. See Sistla 84 for abstract.
- [Clarke 82b] Clarke, E.M. and C.N. Nikolaou.
Distributed Reconfiguration Strategies for Fault Tolerant Multiprocessor Systems.
IEEE Transactions on Computers C-31:771-782, August, 1982.
In this paper, we investigate strategies for dynamically reconfiguring shared memory multiprocessor systems that are subject to common memory faults and unpredictable processor deaths. These strategies aim at determining a *communication page*, i.e. a page of common memory that can be used by a group of processors for

storing crucial common resources such as global locks for synchronization and global data structures for voting algorithms. To ensure system reliability, the reconfiguration strategies must be *distributed* so that each processor *independently* arrives at exactly the same choice. This type of reconfiguration strategy is currently used in the STAGE operating system on the PLURIBUS multiprocessor. We analyze the weak points of the PLURIBUS algorithm and examine alternative strategies satisfying optimization criteria such as maximization of the number of processors and the number of common memory pages in the reconfigured system. We also present a general distributed algorithm which enables the processors in such a system to exchange the local information that is needed to reach a consensus on system reconfiguration.

- [Clarke 83a] Clarke, E.M. Jr., S.M. German, and J.Y. Halpern.
On Effective Axiomatizations of Hoare Logics.
Journal of the ACM 30(3):612-636, July, 1983.

For a wide class of programming languages P and expressive interpretations I , it is shown that there exist sound and relatively complete Hoare logics for both partial-correctness and termination assertions. In fact, under mild assumptions on P and I it is shown that the assertions true in I are uniformly decidable in the theory of $I(\text{Th}(I))$ iff the halting problem for P is decidable for finite interpretations. Moreover the set of true termination assertions is uniformly recursively enumerable in $\text{Th}(I)$ even if the halting problem for P is not decidable for finite interpretations. Since total-correctness assertions coincide with termination assertions for deterministic programming languages, this last result unexpectedly suggests that good axiom systems for total correctness may exist for a wider spectrum of languages than is the case for partial correctness.

- [Clarke 83b] Clarke, E.M., C.N. Nikolaou, N. Francez, and S. Schuman.
A Methodology for Verifying Request Processing Protocols.
In *Proceedings of the ACM SIGCOM '83 Symposium on Communications, Architecture, and Protocols*, Pages 76-83. ACM SIGCOM, March, 1983.

In this paper, we view computer networks as distributed systems that provide their users with a set of services, in a way which hides the distinction between those services which are local and those which are remote. We conceive of a given target network configuration as a network of communicating virtual machines and its behavior is modelled by a system of communicating sequential processes. Network protocols are described by a high level concurrent language (CSP) and a methodology is developed which permits the verification of partial and total correctness assertions about the system in a simple and natural way. Global invariants are used to establish invariant properties of the whole system and histories to record the sequence of communication exchanges between every matching pair of processes. Eventually properties are expressed using linear temporal logic.

- [Clarke 83c] Clarke, E.M., E.A. Emerson, and A.P. Sistla.
Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications.
In *Proceedings of the Tenth ACM Symposium on Principles of Programming Languages*, January, 1983.

We give an efficient procedure for verifying that a finite state concurrent system meets a specification expressed in a (propositional) branching time temporal logic. Our algorithm can be modified to handle *fairness*. We argue that this technique can

provide a practical alternative to manual proof construction or use of a mechanical theorem prover for verifying many finite state concurrent systems.

- [Clarke 83d] Clarke, E. M. Jr. and B. Mishra.
Automatic and Hierarchical Verification of Asynchronous Circuits Using Temporal Logic.
Technical Report CMU-CS-83-155, Carnegie Mellon University Computer Science Department,
1983.
- [Clarke 83e] Clarke, E.M. Jr., E.A. Emerson.
Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons.
Science of Computing(2), July, 1983.
- [Clarke 84a] Clarke, E.M. Jr. and B. Mishra.
Automatic Verification of Asynchronous Circuits,
Lecture Notes in Computer Science. Springer-Verlag Publishers, 1984.
- [Clarke 84b] Clarke, E.M. Jr.
The Characterization Problem for Hoare Logics.
Technical Report CMU-CS-84-109, Carnegie Mellon University Computer Science Department,
1984.

Research by this author and by others has shown that there are natural programming language control structures which are impossible to describe adequately by means of Hoare axioms. Specifically, we have shown that there are control structures for which it is impossible to obtain axiom systems that are sound and relatively complete in the sense of Cook. These constructs include procedures with procedure parameters under standard Algol 60 scope rules and coroutines in a language using recursive procedures without parameters.
- [Clarke 84c] Clarke, E.M. Jr. and A.P. Sistla.
The Complexity of Propositional Linear Temporal Logic.
Journal of the ACM, 1984.
- [Clarke 84d] Clarke, E.M. Jr. and D. Kozen (eds.).
Lecture Notes in Computer Science. Volume 164: *Proceedings of the 1983 Workshop on Logics of Programs.*
Springer-Verlag Publishers, 1984.
- [Clarke 84e] Clarke, E.M. Jr., S.M. German and J.Y. Halpern.
Reasoning about Procedures as Parameters,
Lecture Notes in Computer Science. Springer-Verlag Publishers, 1984.
- [Daniels 83] Daniels, D. and A. Spector.
An Algorithm for Replicated Directories.
Technical Report CMU-CS-83-123, Carnegie Mellon University Computer Science Department,
May, 1983.

This paper describes a replication algorithm for directory objects based upon Gifford's weighted voting for files. The algorithm associated a version number with each possible key on every replica and thereby resolves an ambiguity that arises when directory entries are not stored in every replica. The range of keys associated with

a version number changes dynamically; but in all instances, a separate version number is associated with each entry stored on every replica. The algorithm exhibits favorable availability and concurrency properties. There is no performance penalty for associating a version number with every possible key except on Delete operations, and simulation results show this overhead is small.

- [Durham 82] Durham, I. and M. Shaw.
Specifying Reliability as a Software Attribute.
 Technical Report CMU-CS-82-148, Carnegie Mellon University Computer Science Department,
 December, 1982.
 This paper examines some issues in specifying reliability as a software attribute. A scheme for characterizing software reliability, known as a failure profile, is introduced. Failure profiles are derived for particular implementations of an abstraction by identifying analytically the behavior of the module when software or hardware faults occur. A failure profile is developed for a sorting program to demonstrate an informal techniques for identifying the consequence of faults. The derived failure profile is compared with observations of the program's behavior in the presence of artificially induced faults to demonstrate the effectiveness of the failure profile characterization of software reliability. The issues raised in the application of the informal technique are discussed with respect to developing a formal and more mechanical technique for producing and using failure profiles.
- [Ellison 84] Ellison, R. J.
 The Gandalf Implementation: An Interim Report.
Systems and Software, 1984.
- [Feiler 82] Feiler, P.H.
A Language-Oriented Interactive Programming Environment Based on Compilation Technology.
 PhD thesis, Carnegie Mellon University, May, 1982.
 Also published as CMU-CSD Technical Reports CMU-CS-82-117.
- [Fortune 83] Fortune, S., D. Leivant, and M. O'Donnell.
 The Expressiveness of First and Second Order Type Theories.
Journal of the ACM 30:151-185, 1983.
- [Garlan 84] Garlan, D.B. and P.L. Miller.
 GNOME: An Introductory Programming Environment Based on a Family of Structure Editors.
 In *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, Pages 65-72. April, 1984.
 The Gnome project represents an effort to use the concept of syntax-directed editing in an educational arena. Using a family of editors generated from Medina-Mora's ALOE system yields the added advantage of providing uniform interfaces to each environment used by Gnome students.
- [Goos 81] Goos, G. and W.A. Wulf.
Diana Reference Manual.
 Technical Report CMU-CS-81-101, Carnegie Mellon University Computer Science Department,
 March, 1981.

This document is an introduction to, and reference manual for, Diana, a Descriptive Intermediate Attributed Notation for Ada. Diana is an intermediate form of Ada programs. It is especially suitable for communication between the Front and Back Ends of an Ada compiler, but is also suitable for use with other tools in an Ada programming environment. Diana resulted from a merger of the set properties of two earlier similar intermediate forms: TCOL and AIDA.

[Habermann 81a] Habermann, A.N. and D. Notkin.

The Gandalf Software Development Environment.

In *Proceedings of the Sixth International Conference on Software Engineering*, December, 1981.

Also available in *Proceedings of the Second International Symposium Project on Computation and Information*, September 1983.

An overview of the goals, issues, and approaches of Gandalf are given. The focus of Gandalf on project management, version control, and incremental programming is shown (with less emphasis given to incremental programming since it is described in more detail elsewhere). Also, the motivation for and mechanisms surrounding the environment generation process are stressed. This is more extensive and more current than the overview in the Research Review.

[Habermann 81b]

Habermann, A. N. and D.E. Perry.

System Composition and Version Control for Ada.

In H. Hunke, Editor, *Software Engineering Environment*, Pages 331-343. Gesellschaft für Mathematik und Datenverarbeitung, 1981.

A major part of an integrated programming and system development environment such as Gandalf is that which is concerned with describing systems and controlling versions of those systems. This paper presents the System Version Control Environment (SVCE), discusses motivations that led to its current state, describes a language to depict systems and versions, delineates desirable properties of system component and version descriptions, and illustrates the interaction between SVCE and its users.

[Habermann 81c] Habermann, A. N. and D. Notkin.

An Environment for System Version Control.

In *Proceedings of the Sixth International Software Conference on Software Engineering*, December, 1981.

Also available in *Proceedings of the National Computer Conference*, Jan., 1982. See Habermann 82 for abstract.

[Habermann 82a] Habermann, A.N. and G. Kaiser.

An Environment for System Version Control.

In *Proceedings of the National Computer Conference*, January, 1982.

Also available in *Proceedings of the Sixth International Software Conference on Software Engineering*, Dec. 1981.

This document describes the design of SVCE, an interactive System Version Control Environment. SVCE supports a system description language that describes system and module interfaces and dependencies, a version maintenance facility that handles both parallel and successive versions of modules and systems, and a system generation facility that automatically generates an executable system given the system description. SVCE is a component of Gandalf, an integrated software

development and maintenance environment.

[Habermann 82b]

Habermann, A. N.
Software Development Environment Issues as Related to Ada,
In T. Wasserman, *Software Development Environments*, 1982.

[Habermann 82c] Habermann, A. N.

System Development Environments,
Tools and Notions for Program Construction II. Cambridge University Press, 1982.

[Habermann 83] Habermann, A.N. and D. Perry.

Ada for Experienced Programmers.
Addison Wesley Publishing, 1983.

[Harbison 82a] Harbison, S.P.

The Structure and Communication of Spice Objects.
Spice Project, Carnegie Mellon University Computer Science Department,
April, 1982.

[Harbison 82b] Harbison, S.P.

An Architectural Alternative to Optimizing Compilers.
In *Proceedings of the Symposium on Architectural Support for Programming Languages and Operating Systems*, Pages 57-65. March, 1982.

Programming languages are designed to make *programming* productive. Computer architectures are designed to make *program execution* efficient. Although architectures should be designed with programming languages in mind, it may be as inappropriate to make the computer execute the programming language directly as it is to make the programmer use machine language. It is the compiler's job to match the programming language and the computer architecture, and therefore making compilers efficient and easy to write are important design goals of a complete hardware/software system. This paper summarizes research completed in 1980 on a computer architecture, *TM*, that takes over some of the more burdensome tasks of optimizing compilers for high-level-languages (HLL's), performing these tasks dynamically during the execution of the object program. This is a different approach to making compilers efficient than is commonly taken; more common approaches include devising more efficient optimization algorithms, being clever about when to do optimizations, and building the compilers semi-automatically.

[Hibbard 83]

Hibbard, P.G., A. Hisgen, J. Rosenberg, and M. Sherman.
Studies in Ada Style.
Springer-Verlag Publishers, 1983.

[Kaiser 82a]

Kaiser, G.E, *et al.*
GANDALF Environment Users' Manual and Tutorial.
In *Second Compendium of GANDALF Documentation*, Carnegie Mellon University Computer Science Department, May, 1982.

The manual presents a walk through the GANDALF Prototype from the user's point of view. It describes how the user would perform varied operations, along with a description of the expected (and some unexpected) interactions with the system.

- [Kaiser 82b] Kaiser, G.E and A.N. Habermann.
An Environment for System Version Control.
In *Digest of Papers Spring CompCon '83*, IEEE, November, 1982.
This paper defined the System Version Control Environment that is one of the major portions of the GANDALF Prototype. Motivation and definition of the description and generation process for multiple versions of systems are given.
- [Kaiser 82c] Kaiser, G.E. and E. Kant.
Incremental Expression Parsing for Syntax-Directed Editors.
Technical Report CMU-CS-82-141, Carnegie Mellon University Computer Science Department,
October, 1982.
This paper presents an alternative to dealing with expressions as either pure structure or pure text. The algorithm is based on a set of simple transformations that retain the underlying tree structure of expressions while they are manipulated by the user as tokens.
- [Leivant 81a] Leivant, D.
Implicational Complexity in Intuitionistic Arithmetic.
Journal of Symbolic Logic(46), 1981.
- [Leivant 81b] Leivant, D.
On the Proof Theory of the Modal Logic for Arithmetic Provability.
*Journal of Symbolic Logic*46, 1981.
- [Leivant 83a] Leivant, D.
The Optimality of Induction as an Axiomatization of Arithmetic.
*Journal of Symbolic Logic*48:182-183, 1983.
- [Leivant 83b] Leivant, D.
Polymorphic Type Inference.
Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages:88-98, 1983.
- [Leivant 83c] Leivant, D.
Structural Semantics for Polymorphic Data Types.
Conference Record of the Tenth Annual ACM Symposium on Principles of Programming Languages:155-166, 1983.
- [Leverett 82] Leverett, B.W. and P. Hibbard.
An Adaptive System for Dynamic Storage Allocation.
*Software—Practice and Experience*12, 1982.
- [Medina-Mora 82] Medina-Mora, R.
Syntax-Directed Editing: Towards Integrated Programming Environments.
PhD thesis, Carnegie Mellon University, March, 1982.
Also published as CMU-CSD Technical Report CMU-CS-82-113.
Medina-Mora's thesis studies the issues surrounding the generation of syntax-directed editors that are intended to be the basis of a programming environment (such as the one proposed by Feiler). Restricted template replacement is used to guarantee that all structures, actually trees, are syntactically correct, thereby alleviating the

need for parsing. The ALOE system permits generation of editors for particular languages. Generation of an editor comprises definition of the abstract syntax for a language, the concrete syntax (which indicates how the internal tree is to be displayed to the user), and action routines that define the semantics of the language. Action routines are associated with particular node types and are invoked as modifications, such as creation or deletion, occur at nodes.

[Mishra 83]

Mishra, B. and E.M. Clarke.

Automatic and Hierarchical Verifications of Asynchronous Circuits Using Temporal Logic.

Technical Report CMU-CS-83-155, Carnegie Mellon University Computer Science Department,

September, 1983.

Establishing correctness of complicated asynchronous circuits is in general quite difficult because of the high degree of nondeterminism that is inherent in such devices. Nevertheless, it is also very important in view of the cost involved in design and testing of circuits. We show how to give specifications for circuits in a branching time temporal logic and how to mechanically verify them using a simple and efficient model checker. We also show how to tackle a large and complex circuit by verifying it hierarchically.

[Nestor 81]

Nestor, J.R., W.A. Wulf, and D.A. Lamb.

IDL-Interface Description Language.

Technical Report CMU-CS-81-139, Carnegie Mellon University Computer Science Department,

August, 1981.

IDL is a notation for precisely describing structured data used to communicate among collections of related programs, such as the set of tools in a programming environment. The notation supports the abstract data type paradigm, separating descriptions into abstract properties and concrete properties. Data can be communicated between programs written in different languages on different computer systems via reader/writer utilities which can be generated from the IDL descriptions.

[Notkin 84]

Notkin, D.

Interactive Structure-Oriented Computing.

PhD thesis, Carnegie Mellon University, February, 1984.

Notkin's thesis work views the process of structure-oriented editor generations a general method for constructing general interactive programs. By defining interaction in terms of integration (of data development and program execution), incremental computation, and non-sequential data manipulation, Notkin demonstrates the difficulty of constructing interactive programs given the traditional technology of files, file systems, parsers, text editors, and so on. The Agave system is presented as an alternative environment that interactively supports construction of interactive programs modeled as structure-oriented editors. Extensions to current editor generation systems, particularly in the area of generic operators, are described. Low-level operating system support, based on a two-level capability-based addressing scheme, is also given.

[Perlis 81]

Perlis, A., F. Sayward, and M. Shaw (Eds.).

Software Metrics: An Analysis and Evaluation.

MIT Press, 1981.

- [Reif 83] Reif, J. and W. Scherlis.
Deriving Efficient Graph Algorithms.
Logics of Programs(164), 1983.
- [Rounds 81] Rounds, W. C. and S. D. Brookes.
Possible Futures, Acceptances, Refusals and Communicating Processes.
In *Proceedings of the 22nd Symposium on Foundations of Computer Science*, October, 1981.
- [Scherlis 83a] Scherlis, W.L. and D.S. Scott.
First Steps Towards Inferential Programming.
In *Information Processing 83: Proceedings of the IFIP 9th World Computer Congress*, July, 1983.
Although logics of programs have contributed significantly to our understanding of individual programs and to our knowledge of programming language design, they have had disappointingly little influence on the methods by which programs are constructed and documented in practice. The reason for this, we suspect, is that the understanding embodied in these systems deals with individual programs and does not directly address the process by which programs are constructed. By focusing attention on this process, attempting to discern the fundamental steps in the evolution of programs, we propose that it may be possible to develop a logical system supported by an appropriate machine environment that will be more directly applicable to programming practice. The benefits of such a point of view will be discussed.
- [Scherlis 83b] Scherlis, W.
Software Development and Inferential Programming.
In *Proceedings of the Workshop on Program Transformation and Programming Environments*, November, 1983.
- [Scherlis 84] Scherlis, W.
Applicative and Imperative Programs.
In *Proceedings of the 1984 Workshop on Formal Software Development Combining Specification Methods*, May, 1984.
- [Schwarz 84] Schwarz, P. and A. Spector.
Synchronizing Shared Abstract Types (Revised Issue).
*Transactions on Computer Systems*2(3):223-250, November, 1984.
A formalism for specifying the concurrency properties of such types is developed, based on dependency relations that are defined in terms of an abstract type's operations. The formalism requires that the specification of an abstract type state whether or not cycles involving these relations should be allowed to form. Directories and two types of queues are specified using the technique, and the degree to which concurrency is restricted by type-specific properties is exemplified. The paper also discussed how the specifications of types interact to determine the behavior of transactions. A locking technique is described that permits implementations to make use of type specific information to approach the limits of concurrency.
- [Shaw 81] Shaw, M. (Ed.).
Alphard: Form and Content.
Springer-Verlag Publishers, 1981.

- [Shaw 82] Shaw, M.
Abstract Data Type,
In A. Ralston, *Encyclopedia of Computer Science 2nd. ed.* Van Nostrand Reinhold, 1982.
- [Shaw 83] Shaw, M., E. Borison, M. Horowitz, T. Lane, D. Nichols, and R. Pausch.
Descartes: A Programming-Language Approach to Interactive Display Interfaces.
In *Proceedings of SIGPLAN '83: Symposium on Programming Language Issues in Software Systems*, Pages 100-111. 1983.
- [Shaw 84a] Shaw, M.
The Impact of Modeling and Abstraction Concerns on Modern Programming Languages,
In M. Brodie, J. Mylopoulos and J. Schmidt, *On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Springer-Verlag Publishers, 1984.
- [Shaw 84b] Shaw, M., G.T. Almes, J.M. Newcomer, B.K. Reid and W.A. Wulf.
A Comparison of Programming Languages for Software Engineering,
In A. Feuer and N. Gehani, *Comparing and Assessing Programming Languages: Ada, C, Pascal*. Prentice-Hall, 1984.
- [Sistla 84] Sistla, A.P., E.M. Clarke, N. Francez, A.R. Meyer.
Can Message Buffers Be Axiomatized in Linear Temporal Logic?
In *Information and Control*, Pages 88-95. Oct/Nov, 1984.
Also appeared in *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, Aug., 1982.
Message passing is one of the primary modes of interprocess communication in a distributed system. In this paper we investigate the possibility of *characterizing* and *axiomatizing* different message passing systems in temporal logic. Specifically, we consider FIFO buffers (queues), LIFO buffers (stacks) and unordered buffers (bags). We show that all bounded buffers are characterizable in propositional temporal logic (PTL) and so are axiomatizable. We prove that the theory of unbounded FIFO buffers is π_1^1 -complete and so is not axiomatizable. We also prove that the theories of unbounded LIFO and unordered buffers are decidable and hence are axiomatizable.
- [Spector 83] Spector, A.Z. and P. Schwarz.
Transactions: A Construct for Reliable Distributed Computing.
Operating Systems Review 17(4), April, 1983.
Transactions have proven to be a useful tool for constructing reliable database systems and are likely to be useful in many types of distributed systems. To exploit transactions in a general purpose distributed system, each node can execute a transaction kernel that provides services necessary to support transactions at higher system levels. The transaction model that the kernel supports must permit arbitrary operations on the wide collection of data types used by programmers. New techniques must be developed for specifying the synchronization and recovery properties of abstract types that are used in transactions. Existing mechanisms for synchronization, recovery, deadlock management, and communication are often inadequate to implement these types efficiently, and they must be adapted or replaced.

- [Van Dam 81] Van Dam, A., M. R. Barbacci, C. Halatsis, J. Joosten, M. Letheren.
Simulation of a Horizontal Bit Sliced Processor using the ISPS Architecture Simulation
Facility.
IEEE Computer Society Transactions on Computers C-30(7), July, 1981.

6. Research in Distributed Sensor Networks

A *Distributed Sensor Network* (DSN) is a physically dispersed collection of computers and sensors coupled loosely through a communications network. Network computers monitor a target environment by cooperating to process sensor data. Distributing a task so the processing nodes closest to the sensors can do considerable autonomous processing and decision making presents several challenges. Consider, for example, a network of processors and microphones assigned to track a moving object. At run time, the system must configure itself based on current information about the terrain, knowledge about the object under surveillance, and the state of the network itself. Moreover, in the event of a hardware or software failure in one of its components, the system should be able to reconfigure itself and continue operating with minimal overall performance degradation. Our goal was to design and implement such a system.

During the contract period we designed and implemented the following software systems:

- Accent, a communication-oriented operating system whose primitives support transparent networking, and system reconfiguring and rebinding
- Network interprocess communication protocols that support dynamic rebinding of active communicating computations
- Matchmaker, an interface specification language
- Stardust, a system for dynamic load balancing and fault compensating reconfiguration

The software we built during the contract period has proved valuable in demonstrating key DSN issues. In addition, much of it supports other CMU research projects and we have exported some to external laboratories (e.g., IBM, DEC, and HP) where it has reappeared in commercial applications.

With the aid of our software, we designed and implemented a DSN testbed to demonstrate the feasibility of building systems that meet the needs for self-knowledge about current state, fast, effective error reporting, a capacity for fast localized rebinding, and easy reconfiguration.

6.1. A Communication-Oriented Operating System

Our DSN testbed was implemented using Accent, a network operating system that allows flexible, transparent access to distributed resources. Rashid and Robertson [Rashid 81] developed Accent using an innovative design integrating virtual memory, interprocess communication, and permanent storage. Accent forms the communication-oriented kernel for CMU's Spice operating system (see Chapter 2). We have designed Accent to:

- Provide the ability to create and control numerous independent processes on a single processor that supports interprocess communication.
- Support multiple, independent, virtual address spaces and a virtual machine specification that can

accommodate diverse interpretations of process state.

- Supply two kinds of protection:
 - Address space protection to ensure that no process can affect another except through the interprocess communication facility
 - Access protection in the communication facility itself to prevent unauthorized communication between processes
- Define interprocess communication in a way that allows transparent debugging, monitoring, and fault recovery.
- Take advantage of the debugging, monitoring, and fault recovery mechanisms to allow transparent network extension independent of network hardware or protocols.
- Allow processes to view all services, except the basic communication primitives, as being provided through a communication interface.
- Structure message communication so intermediary processes such as debuggers, protocol converters, or network communication servers, can easily interpret the contents and purpose of messages.

Accent can be viewed as a number of layers, with the system kernel at the bottom and layers of processes providing successively more complex services building upon each other. Interprocess communication through ports provides a uniform interface at each level of the system. Because all system objects and services, including those provided by the kernel, are accessible through messages sent to ports they can be transparently distributed throughout the system.

During the contract period, we implemented a complete Accent version for the Perq Systems Corporation's Perq computer (the initial Spice machine) and deployed it on a network of nearly 150 Perqs within CMU. In addition, we made the Accent IPC facility available on DEC VAXes under a modified version of UNIX4.1bsd. We implemented network interprocess communication servers under both Accent and UNIX and extensively used this IPC facility within the DSN project (e.g. for implementing the testbed) and other DARPA-sponsored projects at CMU particularly Spice.

6.2. Protocols for Network Interprocess Communication

Network interprocess communication protocols enhance system security and support Accent's ability to dynamically rebind active, communicating computations. Implementing Accent demonstrated the value of integrating memory support with interprocess communication.

Accent exemplifies a truly extensible operating system. Accent's port communication concept allows transparent network communication, process monitoring, and debugging without the underlying operating system kernel's intervening or even knowing. Moreover, user-state processes may extend the operating system without changing the underlying kernel. The right to send a message to a port cannot be forged or acciden-

AD-A173 028

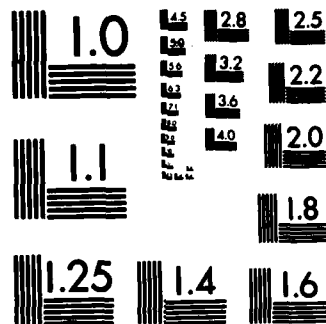
INFORMATION PROCESSING RESEARCH(U) CARNEGIE-MELLON UNIV 2/2
PITTSBURGH PA DEPT OF COMPUTER SCIENCE E BALL ET AL
SEP 86 AFMAL-TR-86-1011 F33615-81-K-1539

UNCLASSIFIED

FFG 9/2

NL

END
12 86
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

tally created since processes only have local references to ports. This prevents either buggy or malicious processes from gaining fraudulent access to resources. Further, it allows a process's author to make a positive statement about his program's correctness, based on precise knowledge of which other processes can communicate with it.

Accent's combined memory/communication provides a clean, kernel-transparent mechanism for cross-network paging whereby one process can manage another's virtual address space and behavior. To accomplish this, Accent can either allocate virtual memory from the kernel, sending it to another process, or explicitly manage page faults. Since ports can be sent in messages to other processes, it is possible for process A to send its kernel port to process B. The process system is designed so that process B can manage process A's behavior, much the same way the virtual memory system allows one process to manage another's virtual memory. This mechanism forms the basis for remote debugging and monitoring systems.

6.3. An Interface Specification Language

One of the thorniest problems in building a distributed system is interfacing the components. Matchmaker [Jones 85] provides an interface specification language for use with existing programming languages and offers:

- A language for specifying object-oriented, remote procedure call (RPC) interfaces between processes executing on the same machine or within the Spice network
- A compiler that converts these specifications into interface code for each of the major languages used within the Spice environment— including C, Perq Pascal, Common Lisp, and Ada—and runtime support for type-checking, communicating, synchronizing, and handling exception.

We began work on Matchmaker in 1981 and first used it to specify the user interface to the Accent operating system kernel. Matchmaker evolved into the effective definition of IPC within the Spice environment and between Perqs and the CS Department's VAXes running our modified UNIX operating system. We have used Matchmaker in the distributed programming support environment for over 500,000 lines of code in four major languages.

6.4. Dynamic Load Balancing and Fault Reconfiguration

Stardust [Hornig 84] is both a language and a system for running programs written in that language. As a language, Stardust provides an applicative language tool for specifying the control structure of large distributed systems such as DSNs. As a system, Stardust provides the runtime support to divide computations into manageable chunks and distribute them throughout the network at runtime. It also can detect error conditions and respond by reconfiguring the system based on its specification. We designed and built Stardust to demonstrate the feasibility of automatic load-balancing and fault recovery in such a network.

We built a Stardust interpreter that successfully handles a variety of distributed system tasks including: the DSN prototype, a molecular modeling task, and classical algorithms (e.g. quicksort, Fibonacci sequence, etc.). The Stardust system successfully demonstrated dynamic load balancing, automatically recovered from failure, and automatically eliminated runtime sub-computation.

6.5. Prototype for Distributed Sensing

With the aid of Accent, IPC protocols, Matchmaker, and Stardust, we built a DSN testbed. The task we chose to demonstrate it was locating a single noise source by analyzing signals received at several microphones. Our strategy was to cross-correlate signal pairs to find the source-sensor time-of-flight difference for each microphone. Each measurement locates the source on a three dimensional quadratic surface. Combining several measurements gives the source coordinates.

Our experimental laboratory included eight microphones located in an acoustically isolated room. We used everyday objects including radios, power drills, and various pre-recorded sounds as subjects for our tracking system. Our computational environment included software running on several VAX-11/780s and Perqs all residing on a 3MHz Ethernet. Data from each microphone was sent in Ethernet packets to the active processors. The number and type of computers could be varied and, in practice, ranged between one and twelve. Data was fed through a series of processes interconnected via the Accent IPC facility. All code was written in Pascal.

We built several prototype systems. In the fall of 1982 we demonstrated an early version that used DPL-82 [Ericson 82], a language specifically written for distributed processing. The system tracked a moving object to within a four inch cube around the sound source. In the summer of 1984, we completed a second system using Stardust. This second system could also track a moving object and displayed greater fault-tolerance in tracking than its predecessors.

6.6. Transaction Based Systems

Building on our work in both distributed sensing and distributed processing, we began developing a Transaction-based System, TABS) [Spector 84a], to explore other reliability issues in distributed systems. Our research focuses on building, atop the existing Accent operating system kernel, a general-purpose, object-oriented transaction mechanism. In TABS, we redefine the "transaction" abstraction, extending it to include sequences of typed operations on objects that themselves are instances of shared data types. The TABS prototype [Schwarz 84] demonstrated the potential of our strategy and guided our continuing research toward higher-performance systems that can more fully exploit available parallelism.

6.7. Bibliography

- [Daniels 83] Daniels, D. and A.Z. Spector.
An Algorithm for Replicated Directories.
 Technical Report CMU-CS-83-123, Carnegie Mellon University Computer Science Department,
 May, 1983.
 This paper describes a replication algorithm for directory objects based upon Gifford's weighted voting for files. The algorithm associates a version number with each possible key on every replica and thereby resolves an ambiguity that arises when directory entries are not stored in every replica. The range of keys associated with a version number changes dynamically; but in all instances, a separate version number is associated with each entry stored on every replica. The algorithm exhibits favorable availability and concurrency properties. There is no performance penalty for associating a version number with every possible key except on Delete operations, and simulation results show this overhead is small.
- [Ericson 82] Ericson, L.W.
DPL-82: A Language for Distributed Processing.
 Technical Report CMU-CS-82-129, Carnegie Mellon University Computer Science Department,
 July, 1982.
 DPL-82 is a language for composing programs of concurrently-executing processes. Processes may be all on a single machine or may be distributed over a set of processors connected to a network. The semantics of the language are derived from the underlying interprocess communication facility (IPC) and from the dataflow model of computation. This paper discusses the major concepts of the language, namely nodes, arcs, connections, tokens, signals, and activations and presents examples which illustrate the construction of distributed programs in DPL-82 with internal arcs, external arcs, and child arcs. Features for process-to-processor mapping and dead process restart are mentioned.
- [Hornig 84] David A. Hornig.
Automatic Partitioning and Scheduling on a Network of Personal Computers.
 PhD thesis, Carnegie Mellon University Computer Science Department, December, 1984.
 Also published as Technical Report CMU-CS-84-165, November 1984.
- [Jones 85] Jones, M.B., R.F. Rashid, and M.R. Thompson.
Matchmaker: An Interface Specification Language for Distributed Processing.
 In *Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 1985.
 Matchmaker, a language used to specify and automate the generation of interprocess communication interfaces, is presented. The process of and reasons for the evolution of Matchmaker are described. Performance and usage statistics are presented. Comparisons are made between Matchmaker and other related systems. Possible future directions are examined.
- [Rashid 81] Rashid, R.F. and G.G. Robertson.
ACCENT: A Communication Oriented Network Operating System Kernel.
 In *8th SOSP Proceedings*, December, 1981.
 Accent is a communication oriented operating system kernel being built at Carnegie-

Mellon University to support the distributed personal computing project, Spice, and the development of a fault-tolerant distributed sensor network (DSN). Accent is built around a single, powerful abstraction of communication between processes, with all kernel functions, such as device access and virtual memory management accessible through messages and distributable throughout a network. In this paper, specific attention is given to system supplied facilities which support transparent network access and fault-tolerant behavior. Many of these facilities are already being provided under a modified version of VAX/UNIX. The Accent system itself is currently being implemented on the Three Rivers Corp. Perq.

- [Rashid 83] Rashid, R.
Accent Kernel Interface Manual.
 Technical Report Spice Project, Carnegie Mellon University Computer Science Department,
 November, 1983.
- [Rashid 84a] Rashid, R.F.
 Accent: A Distributed Operating System for a Network of Scientific Personal Computers.
 In *Proceedings of the Convention Informatique 84*, September, 1984.
- [Rashid 84b] Rashid, R.
 Experiences in the design, implementation and use of network operating systems.
 In *Proceedings of the 1984 Annual Conference of the Italian Computer Society*, October, 1984.
- [Rashid 84c] Rashid, R.
Network Operating Systems.
 Springer-Verlag Publishers, 1984.
- [Schwarz 83] Schwarz, P.M. and A.Z. Spector.
Recovery of Shared Abstract Types.
 Technical Report Spice Project, Carnegie Mellon University Computer Science Department,
 October, 1983.
- [Schwarz 84] Schwarz, P.M.
Transactions on Typed Objects.
 PhD thesis, Carnegie Mellon University Computer Science Department, December, 1984.
 Also published as Technical Report CMU-CS-84-166, December 1984.
- Transactions simplify the construction of reliable systems by protecting programmers from the effects of concurrency and failures. To date, they have been used primarily in database systems. The limitations of existing strategies for synchronization and recovery constrain the use of transactions in more general environments such as file systems, mail systems, and operating systems.
- The standard transaction model can be extended by redefining a transaction as a sequence of typed operations on objects that are instances of shared abstract types. This approach allows semantic knowledge about individual types to be used in developing more flexible synchronization and recovery strategies, including ones that sacrifice basic transaction properties such as serializability. This dissertation discussed a method for specifying the synchronization properties of shared abstract types, and an extensible locking technique for their implementation. This dissertation also examines how a shared abstract type's semantics and implementation

affect the choice of a recovery technique. Two new log-based algorithms for recovery are also presented. The simpler algorithm is easier to implement, and its resource requirements during recovery are more tightly bounded. The more complex algorithm allows greater concurrency in implementations of shared abstract types.

Choosing transactions as a fundamental mechanism for concurrency control and recovery influences the overall structure of a system. The dissertation describes how synchronization and recovery are implemented in TABS, a system being developed at Carnegie Mellon that provides an object-oriented transaction mechanism on top of an existing operating system kernel. The possibility of systems that incorporate transactions at an even lower level is also discussed.

[Sistla 83] Sistla, A.P.
Theoretical Issues in the Design and Verification of Distributed Systems.
 Technical Report CMU-CS-83-146, Carnegie Mellon University Computer Science Department,
 July, 1983.

[Spector 82] Spector, A.Z.
 Performing Remote Operations Efficiently on a Local Network.
Communications of the ACM 25(4), April, 1982.

[Spector 83] Spector, A.Z. and P. Schwarz.
 Transactions: A Construct for Reliable Distributed Computing.
Operating Systems Review 17(4):82-143, April, 1983.
 Also published as CMU-CSD Technical Report CMU-CS-82-143.

Transactions have proven to be a useful tool for constructing reliable database systems and are likely to be useful in many types of distributed systems. To exploit transactions in a general purpose distributed system, each node can execute a transaction kernel that provides services necessary to support transactions at higher system levels. The transaction model that the kernel supports must permit arbitrary operations on the wide collection of data types used by programmers. New techniques must be developed for specifying the synchronization and recovery properties of abstract types that are used in transactions. Existing mechanisms for synchronization, recovery, deadlock management, and communication are often inadequate to implement these types efficiently, and they must be adapted or replaced.

[Spector 84a] Alfred Z. Spector, Jacob Butcher, Dean S. Daniels, Daniel J. Duchamp, Jeffrey L. Eppinger, Charles E. Fineman, Abdelsalam Heddaya, and Peter M. Schwarz.
Support for Distributed Transactions in the TABS Prototype.
 Technical Report CMU-CS-84-132, Carnegie Mellon University Computer Science Department,
 July, 1984.

The Tabs proptotype is an experimental facility that provides operating system-level support for distributed transactions that operate on shared abstract types. It is hoped that the facility will simplify the construction of highly available and distributed applications. This paper describes the TABS system model, the TABS prototype's structure, and certain aspects of its operation. The paper concludes with a discussion of the status of the project and a preliminary evaluation.

[Spector 84b] Spector, A.Z. and D. Gifford (eds.).
The Space Shuttle Onboard System.
Communications of the ACM, September, 1984.

[Spector 84c] Spector, A.Z. .
Software for Process Control.
Scientific American 250(9), September, 1984.

[Spector 84d] Spector, A.Z. and P. Schwarz.
Synchronizing Shared Abstract Types.
Transactions on Computer Systems, August, 1984.
Also published as Technical Report CMU-CS-82-128, July 1982.

This paper discusses the synchronization issues that arise when transaction facilities are extended for use with shared abstract data types. A formalism for specifying the concurrency properties of such types is developed, based on dependency relations that are defined in terms of an abstract type's operations. The formalism requires that the specification of an abstract type state whether or not cycles involving these relations should be allowed to form. Directories and two types of queues are specified using the technique, and the degree to which concurrency is restricted by type-specific properties is exemplified. The paper also discusses how the specification of types interact to determine the behavior of transactions. A locking technique is described that permits implementations to make use of type-specific information to approach the limits of concurrency.

[Spector 84e] Spector, A.Z. and D. Gifford .
The TWA Flight Reservation System.
CACM, July, 1984.

[Spector 84f] Spector, A.Z., J.J. Bloch, and D.S. Daniels.
Weighted Voting for Directories: A Comprehensive Study.
Technical Report CMU-CS-84-114, Carnegie Mellon University Computer Science Department,
April, 1984.

7. Research in Cooperative User Interfaces

Computers provide an ever-expanding range of services to a rapidly growing user pool. Decreasing costs for raw computing power and new computational techniques now encourage designers to shift their primary focus away from programs that use hardware efficiently to programs that let users interact easily with operating systems and application programs. Interacting with most current computer systems remains difficult and frustrating because of rigid, unnatural command languages and poor help facilities. Ironically, interface hardware improvements in the new powerful personal computers have done little to alleviate interface problems. Interfaces using such facilities—e.g. high-resolution bitmap displays and speech processing capabilities—cost much more to implement than those employing simpler technology. Moreover, fancier technology does not guarantee the users will find the end product acceptable.

In the Cooperative User Interface project, our goal is to design, implement, and evaluate powerful interfaces that appear friendly and supportive to users. We also aim to construct our interfaces so many different application programs can use them. During the 1981-84 period, we completed designing and implementing a prototype user interface management system, Cousin, for the Perq Systems Company's Perq, a powerful, bitmapped workstation. Cousin-Spice supports standardized communication between application programs and a centralized user interface program using graphical objects such as forms and menus. We also developed a command interface to the UNIX operating system. Cousin-UNIX provides flexible parsing of a UNIX-style language, interactive error correction of command line errors, and readily accessible help. These interfaces have allowed us to explore directly the following issues inherent in a truly *cooperative* user interface:

- *Robust communication*—an ability to interact with users unhindered by omissions, interjections, and restarts
- *Robust Flexible Parsing*—a way for the computer to fill in contextual information and tolerate syntax deviations by the user.
- *Cooperative error-correction*—ways to shorten users' mistake-recovery time
- *Media-rich communication*—an ability to exploit, for example, high-resolution graphics, voice recognition, and pointing devices
- *Speech and Natural Language Integration*—ways to integrate speech and natural language into interfaces that make man-machine communication more natural for the user
- *Explanation*—ways to make help systems easier to use and to improve users' knowledge of the system's current state.

Graceful interaction with users can be realized using multiple technologies, including: graphics, menus, natural language, speech, well-designed command languages, etc. Here we report on our research in developing natural language interfaces tolerant of errorful, incomplete, or ungrammatical user input, and on its natural extension to the integration of speech and natural language processing using the same basic flexible-

parsing technology.

7.1. Robust Communication

The basic thrust of this area of research is to identify all the information that needs to pass between user and system to accomplish the user's goals, and to develop ways of making this information transfer as efficient and natural for the user as possible.

We developed a contextual interpretation mechanism that follows human intuition closely enough to be natural, but requires no significant cognitive modeling. Our mechanism interprets anaphoric pronouns and noun phrases, elliptical user responses, or contextually dependent, voluntary input. In Cousin-UNIX, the mechanism parses a user command line into an internal "form" representation having slots and values. It provides spelling checks and corrections, phrase completion, and interactive dialogue that obtains information needed to fill the form enough to execute the command. Using information in the environment and the form template, the system dynamically creates dialogue messages to fit the situation [Hayes 85].

Cousin-Spice uses the parsing, spelling checks, and completion, and a similar internal environment representation, but replaces the dialogue with interaction on a graphical form. Defaults, required parameters, etc., are shown in the forms, and users interact by manipulating the objects (buttons, menus, etc.) and values in the forms. Highlighting and locking show omissions or invalid information. By making manipulations "modeless," the user can restart or switch tasks easily at any time. For example, while in the middle of specifying a delete operation, the user could request to look at the files in a different format or order. He could even look at different files or change to a rename command. In Cousin-Spice the context mechanism interprets the user's initial input, but all further interaction and manipulations are in the form and not by natural language.

In order to facilitate efficient and natural information transfer, we designed the Cousin-Spice system to recognize common command sequences by macro-level commands. The system provides appropriate assistance to the user in executing such sequences. Users are able to define their own sequences for repeated use. We also made Cousin capable of multiple-levels of command interaction. Thus users see the same style of interaction with all application packages [Hayes 83a].

7.2. Robust, Flexible Parsing

Users of natural language interfaces seldom observe all the grammatical niceties that schoolteachers insist upon. Nor do users, concentrating on the underlying task, spell out all the implicit aspects to language communication. After all, humans do a fine job at filling in contextual information, and can comprehend

semantically well-formed but syntactically deviant language. Why should not our machines behave similarly? Therefore, our research strives to make machine interfaces equally resilient to variability and minor inaccuracies in the communication medium. To this end, we developed a set of flexible parsers of increasing sophistication, starting from FlexP, and CASPAR to the more recent DYPAR and MULTIPAR systems.

In 1983 we completed DYPAR-II, the first *multi-strategy natural language parser* that unifies syntactic, semantic, and pragmatic constraints, and is capable of ellipsis and anaphora resolution. The DYPAR family of parsers (now through DYPAR-V) have been distributed to hundreds of sites internationally (academic, government and business) and serve as a basis both for academic research projects and potential industrial applications.

Subsequently, we developed the DYPAR line through DYPAR-V, each increasing the sophistication of the former systems, and all exploiting the new case-frame instantiation technology that is proving far superior and more robust than the earlier ATN-based technology. DYPAR-IV and DYPAR-V have been used successfully in building Xcalibur¹, a multi-function natural language interface to expert systems and data bases, and MedSort², a system that extracts indexing information by parsing titles of large numbers of medical texts.

7.3. Cooperative Error-correction

The thrust of our research in this area was based on a conviction, derived from experience with our current system, that efficient error correction requires not only a high degree of interaction, but also error detection, error correction, and ambiguity recognition techniques tailored to the specific error types that can occur. This conviction shifted our emphasis toward more specific techniques for dealing with errors, particularly errors associated with the language and concepts commonly found in the more restricted command interaction domain.

We developed construction-specific, flexible parsing techniques based on our general techniques for flexible parsing. These specific techniques allow us to exploit the different roles and ease of recognition of the various constituents of each type of construction in the languages we will deal with. For example, if the arguments to a command (cases of a verb) are flagged by keywords (prepositions), strategies for dealing with grammatically deviant uses of that command (verb phrase) can make use of the fact that the keywords are much easier to recognize than the parameters (deep cases) they mark. Keywords, for instance, can be scanned to "realign" a parse thrown off by missing or extraneous constituents. We also determined the specific types

¹Xcalibur was a DEC-sponsored project

²MedSort was an NIH-sponsored project

of ambiguity that each construction cause, developed formalisms for representing each of these ambiguities without duplicating unambiguous constituents, and ensured that our construction-specific parsing techniques can make use of these representations to report ambiguities. In addition, we developed methods for the system to ask intelligently directed questions to resolve ambiguities thus represented, and to understand appropriately economical replies from the user [Hayes 82a]. We used these parsing techniques for the initial input in Cousin-Spice and used them extensively in Cousin-UNIX.

At the lexical level, we integrated the spelling correction techniques already developed with techniques for dealing with abbreviations, inflections, and unknown proper names (of files, other users, etc.). We also fully integrated all aspects of parsing with other parts of the system, so that, for instance, spelling correction will interact intelligently with the semantic tests of command checking.

A command-checking mechanism common to both Cousin-Spice and Cousin-UNIX is now in place. It deals with parameter correctness, default provisions, parameter ambiguity resolution, etc. Cousin-Spice is broken into two major components, the Form-Manager and the Form-Editor. The Form-Manager embodies most of the facilities in Cousin-UNIX, such as the checking, dealing with ambiguities, and parsing. The Form-Editor handles the interaction between the user and the Manager by means of a graphical display of the form kept in the Manager and an editor which allows the user to modify that form [Hayes 84]. The low-level mechanisms are very similar in both Cousin-UNIX and Cousin-Spice.

We investigated ways of displaying error messages from subsystems invoked by Cousin. Initially, little processing was performed on these messages, but as the system implementation progressed, we devised methods of relating errors much more closely to the context of the preceding interaction with the interface [Hayes 83a].

Most of the error messages and subsequent dialogue with the user, which is present in Cousin-UNIX, is not needed in Cousin-Spice. Whereas, in Cousin-UNIX, the state of the interaction is kept internally and made available to the user by a dialogue, in Cousin-Spice, virtually the entire state is explicitly shown to the user. Errors are obvious to the user by sight, or in some cases, through the use of menu selection: The user cannot make a mistake. For example, to list the contents of a directory, using Chili, it would be impossible for the user to select a non-editing flag since only the valid flags are shown.

7.4. Media-rich Communication

Interactions restricted to typed commands and character output represent a major source of difficulties encountered with most current computer systems. Our research in media-rich communications emphasizes increasing efficiency at the man-machine interface by developing techniques that exploit the capabilities of high-resolution graphics displays, voice recognition, and pointing devices.

For Cousin-Spice, we designed and implemented a graphics support package in Pascal using Spice Sapphire. This provided powerful facilities for manipulating a high-resolution raster display without introducing dependencies on display resolution or pixel representation. The package allows concurrent access to a single display device by multiple processes and provides convenient mechanisms for the allocation of display resources.

We constructed a set of widely applicable facilities for generating effective displays of information within the system. The display capabilities include: curve plotting, bar graph generation, and histogram displays. These utilities are available for use within Cousin-Spice and by applications subsystems to present information to the user in an effective manner. We also designed a language for defining data filters. A filter specifies the mapping from an internal data representation to the graphics display. Multiple filters allow the same internal information to be presented to the user in different ways.

CUI researchers designed Cousin-Spice's forms and menus; they specified the display format, type, and (optionally) default value for each field. At this point, applications designers can interface their subsystems with Cousin. The user interaction facilities available in the system make extensive use of application-specific information provided by the tool designer in declarative form [Hayes 83a]. The Form-Editor provides the display. The SDO Editor and the Layout Editor let the application programmer describe an application and how it should be displayed to the user. Another program generates "help" files from application descriptions. Cousin-UNIX has a similar program.

7.5. Integrating Speech and Natural Language

The successful integration of speech recognition and natural language processing technologies promises substantial benefits, but poses some serious challenges. The integration problems require a computational solution in which the natural language component is based on robust case-frame analysis and island-growing techniques. We based our approach on the best aspects of DYPAR and MULTIPAR, our earlier natural language systems which are superior to methods in which the natural language component is based on transition networks (e.g. ATNs).

In 1981 we designed and implemented a speech recognition system integrated with a mail system. The system allowed a user to read, generate, and send messages, controlling the process and receiving instructions by voice. The voice message system used the Lincoln Lab LPC voice encoder for input and output and employed a template-based recognizer. The recognizer was partially microcoded on a Perq and had a window- and menu-based interface (voice substituted for the mouse). Although speed and accuracy allowed the system to be effectively demonstrated, we believe a higher accuracy than the 6% achieved is necessary to effectively perform such a task. The system was not extended since it was felt that better recognition algorithms were necessary.

7.6. Explanation

In this area, we are concerned with fulfilling the user's need to know about the system in general: what commands it will accept and how those commands must be stated (static explanation). Cousin-UNIX explains system requirements (drawn from the tool description) through dialogue with the user [Glasner 81]. Cousin-Spice automatically generates static explanations through form highlighting [Hayes 82b].

The user will also want information about specific states of the interaction: what the system is currently doing, why it is doing it, what it expects the user to do, whether certain events have happened, what is the nature of their outcome. A help system was built for Cousin-Spice which used the context of the current form to explain the current state of interaction [Hayes 82b], such as listing the fields which are incorrect and what is wrong with them.

7.7. Bibliography

- [Ball 82] Ball, J.E. and P.J. Hayes.
A Test-bed for User Interface Designs.
In *Proceedings of the Conference on Human Factors in Computer Systems*, National Bureau of Standards, March, 1982.
- [Carbonell 84] Carbonell, J.G. and P.J. Hayes.
Recovery Strategies for Parsing Extragrammatical Language.
Technical Report CMU-CS-84-107, Carnegie Mellon University, Computer Science Department,
February, 1984.
Practical natural language must exhibit robust behavior in the presence of extragrammatical user input. This paper classifies different types of grammatical deviations and related phenomena at the lexical, sentential, and dialogue levels, and presents recovery strategies tailored to specific phenomena in the classification. These strategies constitute a tool chest of computationally tractable methods for coping with extragrammaticality. Some of the strategies have been tested and proven viable in existing parsers.
- [Glasner 81] Glasner, I.D. and P.J. Hayes.
Automatic Construction of Explanation Networks for a Cooperative User Interface.
Technical Report CMU-CS-81-146, Carnegie Mellon University, Computer Science Department,
August, 1981.
This paper is concerned with providing *automatically generated on-line explanations* to the user of a functional computer subsystem or *tool* about what the tool can and cannot do, what parameters and options are available or required with a given command, etc. The explanations are given through the COUSIN interface system which provides a cooperative tool-independent user interface for tools whose objects, operations, input syntax, display formats, etc. are declaratively represented in a *tool description* data base. The explanations are produced automatically from this data base, with no incremental effort on the part of the tool designer, and in a single uniform style for any tool that uses COUSIN as its interface. The explanation facility takes the form of a fine-grained, tightly linked network of text frames supported by the ZOG menu-selection system. Exactly what information the net building program, NB, extracts from a tool description, and the way in which this information is formatted in the text frames is controlled by a second declarative data base called the *aspect description*. The declarative nature of the aspect description makes it easy to adapt NB to changes in and extensions to the tool description formalism, and to experiment with the structure of the explanation network. We also describe how the appropriate network frame can be found and displayed in response to specific explanation requests from the user.
- [Hayes 82a] Hayes, P.J.
Cooperative Command Interaction through the COUSIN System.
In *Proceedings of the International Conference on Man/Machine Systems*, Institution of Electrical Engineers, University of Manchester Institute of Science and Technology,
July, 1982.
Many of today's interactive interfaces to computer systems are sources of great frustration to their users. The simplest error or incompleteness in a command to such a

system is likely to elicit a more or less informative error message and a request to try again. Different parts of the same interface may use quite different syntax or conventions for essentially similar functions. The online help, if it exists, may come in chunks too big to be useful for interactive use, and may be indexed and cross-referenced inadequately to permit easy location of the information desired. These and other problems with interactive interfaces have been discussed at length numerous authors. In the COUSIN (Cooperative user interface) project at Carnegie-Mellon University, we are working towards user interfaces that appear more friendly and supportive to their users, thus reducing frustration and enhancing productivity. This paper will describe our current work on interactive operating system command interfaces.

[Hayes 82b]

Hayes, P.J.

Uniform Help Facilities for a Cooperative User Interface.

In *Proceedings of the National Computer Conference*, AFIPS, June, 1982.

This paper describes the design of the help and explanation component of a user-friendly operating system command interface called Cousin. The facility can provide two kinds of information: (1) static descriptions of the various subsystems that can be invoked, their parameters, and the syntax that must be used; (2) dynamically generated descriptions of the state of the current interaction, why that state has arisen, and what the user's options for action are. Both types of information are presented in the same way through a network of small text frames connected by semantically motivated links in the style of the Zog system. Frames containing static information are generated automatically for each subsystem from a declarative description of the subsystem which is also used by Cousin for its other services, including spelling and grammar correction, and interactive error resolution. Dynamically generated frames are incorporated temporarily into the static network with semantic links appropriate to the current command context.

[Hayes 83a]

Hayes, P.J. and Szekely, P.A.

Graceful Interaction Through the Cousin Command Interface.

*International Journal of Man-Machine Studies*19(3):285-305, September, 1983.

Currently available interactive command interfaces often fail to provide adequate error correction or on-line help facilities, leading to the perception of an unfriendly interface and consequent frustration and reduced productivity on the part of the user. The Cousin project of Carnegie-Mellon University is developing command interfaces which appear more friendly and supportive to their users, using a form-based model of communication, and incorporating error correction and on-line help. Because of the time and effort involved in constructing truly user-friendly interfaces, we are working on interface systems designed to provide interfaces to many different application systems, as opposed to separate interfaces to individual applications. A Cousin interface system gets the information it needs to provide these services for a given application from a declarative description of that application's communication needs.

[Hayes 83b]

Hayes, P.J. and D.R. Reddy.

Steps Toward Graceful Interaction in Spoken and Written Man-Machine Communication.

*International Journal of Man-Machine Studies*19(3):211-284, September, 1983.

[Hayes 84]

Hayes, P.J.

Executable Interface Definitions Using Form-Based Interface Abstractions,

In H.R. Hartson, *Advances in Computer-Human Interaction*. Ablex, 1984.

The integral bit-map displays and considerable computational power of the new generation of personal workstations offer the possibility of excellent user interfaces. Yet this potential is often unfulfilled because of the cost and complexity of building user interfaces that fully exploit the available resources. A solution to this problem is to define user interfaces through a language embodying appropriate interface abstractions. Such interface definitions can be interpreted by a central interface system to realize an interface that a user can interact with. If the interface abstractions employed are at a suitably high level, the task of constructing individual interfaces is much simplified, with the complexities of exploiting sophisticated interface hardware limited to the construction of the central interface system.

A specific set of interface abstractions is presented. The abstractions are oriented around a form-filling metaphor of communication between user and application program. They are suitable for defining command interfaces for many, but not all, applications. An attempt is made to delimit their range of applicability.

An interface system that runs on a Perq, a powerful personal workstation, is described. This interface system can interpret interface definitions expressed in a language embodying the interface abstractions just mentioned. The result of this interpretation is a graphical interface with many user-friendly features. An example of an interface description definition and the interface that results from it is given.

[Hayes 85]

Hayes, P.J., P.A. Szekely, and R.A. Lerner.

Design Alternative for User Interface Management Systems Based on Experience with Cousin.

In *CHI '85 Proceedings*, April, 1985.

User interface management systems (UIMSs) provide user interfaces to application systems based on an abstract definition of the interface required. This approach can provide higher-quality interfaces at a lower construction cost. In this paper we consider three design choices for UIMSs which critically affect the quality of the user interfaces built with a UIMS, and the cost of constructing the interfaces. The choices are examined in terms of a general model of a UIMS. They concern the sharing of control between the UIMS and the application it provides interfaces to, the level of abstraction in the definition of the sequencing of the dialogue. For each choice, we argue for a specific alternative. We go on to present Cousin, a UIMS that provides graphical interfaces for a variety of applications based on highly abstracted interface definitions. Cousin's design corresponds to the alternative we argued for in two out of three cases, and partially satisfies the third. An interface developed through, and run by Cousin is described in some detail.

8. Research in Integrated VLSI Systems

Very large scale integration (VLSI) technology promises to profoundly change the face of computing by providing cheap and massively parallel machines. We face three challenges in realizing this promise: taming the complexity of designing systems with billions of transistors, understanding how technology affects architecture and how architecture changes with technology, and learning how to design highly parallel machines that make efficient use of hardware resources and integrate them into application systems.

Our activities during the contract period, all motivated by these themes, fall into three general classes: theory, systems and design tools.

- *VLSI Theory*—Traditional models of computational complexity are incapable of capturing the behavior of large VLSI systems. First, new issues arise because of the availability of massive parallelism, and issues of communication and synchronization become central. Second, new hardware constraints bring technology issues into greater prominence: a central example is that although traditional models stress the cost of gates and de-emphasize the cost of interconnection, exactly the opposite is true for VLSI.

The VLSI theory effort at CMU prior to 1981 pioneered new models of computational complexity and effective parallel architecture design. This included the formulation of the concept of *systolic algorithms*, which allow the creation of very high performance, yet simple to build, VLSI systems. Since 1981, we have made great progress in understanding the design of systolic algorithms and in applying theoretical tools to previously ill-formed implementation issues such as wafer scale integration, problem decomposition for special-purpose processors, and global synchronization.

- *VLSI Systems*—Beginning in 1981, our VLSI systems work has had two thrusts: developing parallel architectures for specific application areas, and designing custom chips and integrating them into realistic working systems. The architecture work stresses the analysis and exploitation of application and algorithm features to allow the design of highly efficient parallel architectures whose communication, computation and control resources are closely tailored to the problem. Our hardware work has concentrated on building our expertise in chip and system design, proving architectural concepts, and gaining experience that will guide us in further theoretical and practical work.
- *VLSI Design Tools*—For designs of the tremendous complexity provided by VLSI technology to be feasible, they must be supported by sophisticated computer aided design tools. This is especially true for special purpose systems, whose design cost represents a large portion of system cost due to low volume. Our work on design tools, concentrating largely on design validation problems, has resulted in a number of novel tools and algorithms, both powerful and CPU-efficient, that have aided our own design work and have been distributed to hundreds of other sites.

8.1. VLSI Theory

Over the past four years, our VLSI theory efforts have been increasingly guided by our experience in the design of algorithms, chips, and systems. Accordingly, our results have been oriented more and more toward using theoretical tools to gain insight into ill-structured real-world problems. Our results fall into three main

areas:

- *Algorithms*—We have continued our work on developing individual systolic algorithms and understanding their properties. We have also worked on algorithms that do not fit neatly into the systolic framework, but that share similar features.
- *Theory of algorithm design*—In an effort to provide understanding of and tools for the design of systolic algorithms, we have developed a number of approaches. We have developed both mathematical and intuitive tools for describing and manipulating systolic algorithms.
- *Implementation issues*—As we and other groups gain experience in VLSI system design, various implementation issues arise for which theoretical analysis is valuable. We have carried out theoretical investigations of global synchronization, wafer scale integration and fault tolerance, and the problems involved in partitioning large problems to run on a fixed-size special purpose device.

8.1.1. Algorithms

While gaining a greater understanding of the systolic approach to parallel computing and building actual systolic prototypes, we have designed a large number of individual systolic and near-systolic algorithms. These include algorithms for the following problems:

- polynomial GCD [Brent 82a, Brent 84a]
- integer GCD [Brent 83a, Brent 84b]
- *computational geometry problems* [Chazelle 82]
- median and related filtering [Fisher 82a, Ofllazer 83]
- dictionary problems [Fisher 84a]
- regular language recognition [Foster 82, Foster 83, Foster 84]
- image convolution [Kung 82a, Kung 83a, Kung 84a]
- computational algebra [Kung 83b]

8.1.2. Theory of algorithm design

Since parallel algorithms are often harder to design, understand, and prove correct than serial algorithms, the question of formal systems for manipulating systolic algorithms has received much attention. Two key contributions were made at CMU. Lam and Mostow [Lam 83] have produced a prototype system, based on program transformation principles, that helps a user transform a serial program into a systolic parallel program. Kung and Lin [Kung 84b, Kung 83c] have developed an algebra for manipulating and proving correctness properties of systolic algorithms.

8.1.3. Implementation issues

An important issue in implementing special-purpose hardware is how to decompose problems that are too large to be solved in a single pass. Hong and Kung [Hong 81] have developed a model for the I/O costs of problem decomposition, based on an abstract pebbling game. Using this formalism, they derived lower bounds on communication costs depending on algorithm properties. In a more applied vein, Kung and Yu [Kung 82b] have explored some of the practical issues of problem decomposition on a machine incorporating multiple special-purpose devices.

Another implementation issue, and one that assumes increasing importance as systems include more processors, is synchronization. As systems become more complex, global synchronization by means of a broadcast clock signal becomes more difficult to implement. Fisher and Kung [Fisher 83a] have developed a model of the time costs of global synchronization, and have derived upper and lower bounds for clocking different array topologies under different assumptions about clock skew. A typical result is that two-dimensional arrays are inherently harder to clock than one-dimensional arrays.

A third implementation issue that we have examined is Wafer scale integration of VLSI processor arrays. Using an entire silicon wafer for a system provides obvious density advantages, but necessitates the use of redundancy and fault tolerant design to achieve acceptable yield. Kung and Lam [Kung 83d, Kung 84c] have shown how the delays introduced by faulty elements of an array can be "hidden" in the normal timing scheme of a systolic array, and have performed simulation studies producing wafer yield estimates for varying defect densities.

8.2. VLSI Systems

Our VLSI systems work continues to emphasize the development of task-specific architectures and chips that serve as system building blocks. The areas we have developed architectures in include database management, production systems for artificial intelligence, speech understanding, and systolic processing. We have developed two building block chips for systolic arrays, along with a multi-purpose board level systolic processor. We have also built chips for a number of specific applications, including move generation for a high-performance chess playing machine. We highlight a few examples below.

8.2.1. Architectures

During the 1981-84 period, our work on implementing systolic arrays shifted from single-purpose prototypes to programmable architectures capable of serving many applications. Our first such endeavor was the design, implementation, and system integration of the programmable systolic chip described in 8.2.2. This guided the design of a successor, the Warp processor (which has subsequently been constructed under the

auspices of the DARPA Strategic Computing program) and a related interconnection chip, the LINC(described below).

In collaboration with the artificial intelligence group at CMU, we have been pursuing the design of efficient parallel architectures for interpreting production systems. We began by analyzing a set of existing production systems and evaluating the potential speedup available through parallel implementation. This study [Gupta 84a] concluded that massive parallelism would not be helpful in executing OPS5-like production systems. Using these results, two parallel architectures have been developed: Gupta [Gupta 84b] has proposed a bus-based architecture, and Oflazer [Oflazer 84] has proposed a tree-structured architecture. Further studies continue under the DARPA Strategic Computing project.

Another application we have examined is parallel processing of large databases. We have carried out three studies with varying emphasis. Song [Song 81a] proposed a tree-structured machine for backend database processing and studied its properties. Lehman [Lehman 81] developed a systolic system for database transaction processing. Oflazer carried out a design study for a VLSI reimplementation of an existing parallel database machine.

We have produced and evaluated various designs of custom systems suitable for beam search algorithms. After recognizing that no programmable machine could attain the desired cost/performance ratio and that many components of beam search (e.g. the heuristic pruning function) should be programmable, we decided to build a set of tools that would help automate the design of custom systems for beam search. We used a self-timed architecture which freed us from the requirement of finely tuned hardware because it could correctly interconnect components of variable speed. We successfully designed, fabricated, and tested a number of basic components: latches, adders, and multipliers. We designed a set of tools that translates an algorithm's graphical representation into a wirelist of basic components. Although the graphical description proved inconvenient, we identified a number of optimizations of the algorithm for silicon implementation. Our initial desire was to build enough tools to be able to close the design loop only after having a realistic layout of a chip, but we could not build a satisfactory layout estimation tool because of the complexity and manpower requirements of the task. We did design and build a time analyzer which could identify the delay between the input and different parts of the design when the system was in steady state. This tool allowed us to identify the approximate performance of the design and to place pipeline latches in the best way.

8.2.2. Chips

Our first foray into the construction of programmable systolic arrays as well as the design of complex chips was the programmable systolic chip

(PSC) [Fisher 84b, Kung 83e]. Conceived at the end of 1981 and reduced to silicon a year later, the PSC is a single-chip microprocessor designed to be used in groups to implement high-performance systolic arrays. Its novel structure is tailored to the computation, communication and control requirements of systolic algorithms. The PSC was fabricated by MOSIS, and an image processing array built from PSCs has been demonstrated.

In conjunction with the design of the Warp processor, a board-level follow-on to the PSC, Hsu, Kung and Sussman have designed LINC [Hsu 84], a high-performance interconnection chip. A LINC (Link and Interconnect Chip) includes a crossbar and delay elements, and serves as the communication medium among the functional elements of a single high-performance processing element. The LINC is being fabricated by General Electric in 1.2 micron CMOS.

An especially successful chip has been designed by Ebeling [Ebeling 84a] for use in a chess machine. The chip computes, for a given game position and given square of a chess board, all possible legal moves to that square. Thus 64 chips can be used in parallel to generate all possible moves from any one position. The chip also contains support for the usual search algorithms used in game playing. Ebeling and Slomer have constructed a special-purpose machine incorporating 64 custom chips and a bit-slice controller. The resulting system has attained the highest rating ever held by a chess program, and recently won the 1985 ACM computer chess championship.

Foster developed a series of chips implementing regular expression matching algorithms, furthering his and Kung's work on pattern recognition algorithms. The particularly interesting feature of these chips is that rather than being specialized for particular expressions at the mask level, they were designed for customization by laser disconnection of metal wires [Foster 83, Foster 82]. The chips were fabricated by MOSIS, and successfully programmed and tested in collaboration with MIT Lincoln Laboratories.

Raibert and his colleagues have exploited the physical, as well as the electronic, properties of silicon chips to build a series of novel tactile sensor chips [Raibert 82a, Raibert 82b]. The basic idea is to leave windows in the chip's protective glass coating, allowing electrical contact with metal wires on the chip, and to cover the chip with a conductive elastomer. Different pressures on the chip surface then result in different current flows among the electrical nodes on the chip. On-chip processing aids in the analysis of the pressure data thus obtained.

A more applied example of our chip-building experiments was the design and use of a custom chip for testing custom chips. Ebeling and Frank designed a chip for sending and receiving large test vectors under microprocessor control, and they and Anantharaman constructed a tester using a group of MOSIS chips. The tester has been in use for several years and has been used on many later designs.

8.3. VLSI Design Tools

We have centered our VLSI design tool work on the themes of analyzing the needs of designers, and then analyzing the properties of typical design databases to determine efficient algorithms. This approach has resulted in a number of tools that give great leverage in the design of complex systems, yet run with great resource efficiency. In particular, we have implemented and distributed a pair of circuit extractors, one flat and one hierarchical, a circuit comparator, and a tool that evaluates the yield characteristics of a layout under various defect density assumptions. To support our own system building efforts, we have also constructed a powerful circuit board design system and a suite of chip-testing software.

One of the basic steps in current VLSI design practice is to extract an abstract circuit representation of a given chip layout. Gupta's ACE [Gupta 83a], the first of two high performance circuit extractors written at CMU, has been distributed to hundreds of academic and industrial sites, and is one of the fastest "flat" extractors in existence. Following ACE's success, Gupta and Hon [Gupta 83b, Hon 83] collaborated to produce a second extractor, HEXT, that exploits the hierarchical structure of a VLSI layout to reduce processing time.

A particularly helpful way to use the extracted layout is to compare it with a circuit specification generated either by a netlist language or a schematic capture system. Ebeling designed and implemented a hashing-based graph matching algorithm that performs this task very rapidly. His program, Gemini [Ebeling 83], is in use at a large number of labs outside CMU. Locally, its use has measurably improved our chip design productivity by eliminating a major class of layout errors.

Another facet of layout analysis is yield estimation. Walker's VLASIC [Walker 83a] is an integrated circuit yield simulator that models the functional yield loss caused by local process faults such as oxide pinholes, extra and missing material, and junction leakage. The simulator uses a Monte Carlo process of placing defects on a layout according to the statistics observed in the fabrication line and analyzing them to determine what circuit faults have occurred. VLASIC can function as a component in application systems, such as a statistical design rule developer, an inductive fault analyzer, or a redundancy analyzer. The Semiconductor Research Corporation also supports this work.

We have also developed two tools targeted more for practical utility than for research value. The first is a

printed circuit board design package that guarantees correctness of connectivity and design rules and is also capable of automatic routing. The second is a software package [labeling 84b] for our custom chip-based chip tester. Both tools have seen heavy use in our experimental work.

8.4. Bibliography

[Anantharaman 84]

Anantharaman, T., M. Annaratone, and R. Bisiani.
A Family of Custom VLSI Circuits for Speech Recognition.
In *ICASSP '84*, IEEE, March, 1984.

[Annaratone 84a] Annaratone, M.

Let's Design CMOS Circuits! - Part One.
Technical Report CMU-CS-84-101, Carnegie Mellon University Computer Science Department,
January, 1984.

[Annaratone 84b] Annaratone, M. and W.Z. Shen.

The Design of an LSI Booth Multiplier: nMOS vs. CMOS Technology.
Technical Report CMU-CS-84-150, Carnegie Mellon University Computer Science Department,
September, 1984.

[Bentley 81]

Bentley, J. and T. Ottmann.
The Complexity of Manipulating Hierarchically Defined Sets of Rectangles.
Technical Report CMU-CS-81-109, Carnegie Mellon University Computer Science Department,
April, 1981.
Also available in Proceedings: Tenth International Symposium on the Mathematical Foundations of Computer Science, August 1981.

[Bentley 82]

Bentley, J.L.
The Interaction of VLSI Theory and Practice: A Case Study.
In *Proceedings of the Conference on Advanced Research in VLSI*, January, 1982.

[Bentley 83]

Bentley, J.L. and H.T. Kung.
An Introduction to Systolic Algorithms and Architectures.
*Naval Research Reviews*XXXV(Two):3-16, 1983.

[Bisiani 82a]

Bisiani, R. and A. Waibel.
Performance Trade-offs in Search Techniques for Isolated Word Speech Recognition.
In *International Conference on Acoustics, Speech and Signal Processing*, IEEE-ASSP, May, 1982.

[Bisiani 82b]

Bisiani, B., M.J. Foster, H.T. Kung, and K. Oflazer.
MISE: Machine for In-System Evaluation of Custom VLSI Chips for Real-Time Systems.
Technical Report CMU-CS-82-132, Carnegie Mellon University Computer Science Department,
September, 1982.
Also in 'Proceedings of Real-Time Systems Symposium,' Dec 1982.

[Bisiani 83a]

Bisiani, R., H. Mauersberg, and R. Reddy.
Task-Oriented Architectures.
Proceedings of the IEEE, July, 1983.

- [Bisiani 83b] Bisiani, R., M. Annaratone, and M.J. Foster.
An Architecture for Real Time Debugging of Custom VLSI Chips.
In *1983 International Symposium on VLSI Technology, Systems and Applications*, March, 1983.
- [Bisiani 83c] Bisiani, R.
A Class of Data-flow Architectures for Speech Recognition.
In *Proceedings of the 1983 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, April, 1983.
- [Bisiani 83d] Bisiani, R. and R. Reddy.
Task-Oriented Architectures.
In *Proceedings of the IEEE*, IEEE, July, 1983.
- [Bisiani 84] Anantharaman, T., M. Annaratone and R. Bisiani.
A Family of Custom VLSI Circuits for Speech Recognition.
In *IEEE International Conference on Acoustics, Speech and Signal Processing*, March, 1984.
- [Bojanczyk 81] Bojanczyk, A., R.P. Brent, and H.T. Kung.
Numerically Stable Solution of Dense Systems of Linear Equations Using Mesh-Connected Processors.
Technical Report CMU-CS-81-119, Carnegie Mellon University Computer Science Department,
January, 1981.
- [Bojanczyk 84] Bojanczyk, A., R.P. Brent and H.T. Kung.
Numerically Stable Solution of Dense Systems of Linear Equations Using Mesh-Connected Processors.
SIAM Journal on Scientific and Statistical Computing 5(1):95-104, March, 1984.
- [Brent 82a] Brent, R.P. and H.T. Kung.
Systolic VLSI Arrays for Polynomial GCD Computation.
Technical Report CMU-CS-82-118, Carnegie Mellon University Computer Science Department,
May, 1982.
See Brent and Kung Aug. 1984, *IEEE Transactions on Computers* c-33(8) for abstract.
- [Brent 82b] Brent, R.P. and H.T. Kung.
A Regular Layout for Parallel Adders.
In *IEEE Transactions on Computers*, Pages 260-264. IEEE, March, 1982.
- [Brent 83a] Brent, R.P. and H.T. Kung.
Systolic VLSI Arrays for Linear-Time GCD Computation,
In Anceau, F. and E.J. Aas, *VLSI '83*, Pages 145-154. North-Holland, 1983.
The problem of finding a greatest common divisor (GCD) of any two nonzero polynomials is fundamental to algebraic and symbolic computations, as well as to the decoder implementation for a variety of error-correcting codes. This paper describes new systolic arrays that can lead to efficient hardware solutions to both the GCD problem and the extended GCD problem. The systolic arrays have been implemented on the CMU programmable systolic chip (PSC) to demonstrate its application to the decoder implementation for Reed-Soloman codes. The integer GCD problem is also considered, and it is shown that a linear systolic array of $O(n)$

cells can compute the GCD of two n -bit integers in time $O(n)$.

- [Brent 83b] Brent, R.P., H.T. Kung, and F.T. Luk.
Some Linear-Time Algorithms for Systolic Arrays.
In *Proceedings of the IFIP 9th World Computer Congress*, Pages 865-876. IFIP, September, 1983.
- [Brent 84a] Brent, R.P. and H.T. Kung.
Systolic VLSI Arrays for Polynomial GCD Computation.
IEEE Trans on Computers C-33(8):731-736, August, 1984.
The problem of finding a *greatest common divisor* (GCD) of any two nonzero polynomials is fundamental to algebraic and symbolic computations, as well as to the decoder implementation for a variety of error-correcting codes. This paper describes new systolic arrays that can lead to efficient VLSI solutions to both the GCD problem and the extended GCD problem.
- [Brent 84b] Brent, R.P. and H.T. Kung.
A Systolic Algorithm for Integer GCD Computation.
Technical Report CMU-CS-84-135, Carnegie Mellon University Computer Science Department, June, 1984.
Also appeared in *Transactions on Computers*.
We show that the greatest common divisor of two n -bit integers (given in the the usual binary representation) can be computed in time $O(n)$ on a linear array of $O(n)$ identical systolic cells, each of which is a finite-state machine with connections to its nearest neighbors.
- [Chazelle 81a] Chazelle, B.M. and L.M. Monier.
Optimality in VLSI.
In *First International Conference on Very Large Scale Integration*, Pages 269-278. August, 1981.
- [Chazelle 81b] Chazelle, B.M. and L.M. Monier.
Unbounded Hardware is Equivalent to Deterministic Turing Machines.
In *First Conference on Foundations of Software Technology and Theoretical Computer Science*, December, 1981.
- [Chazelle 81c] Chazelle, B.M. and L.M. Monier.
A Model of Computation for VLSI with Related Complexity Results.
In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, Pages 318-325. ACM SIGACT, May, 1981.
- [Chazelle 81d] Chazelle, B.M. and L.M. Monier.
Towards More Realistic Models of Computation for VLSI.
In *Proceedings of the Second CalTech VLSI Conference*, California Institute of Technology, January, 1981.
- [Chazelle 82] Chazelle, B.
Computational Geometry on a Systolic Chip.
Technical Report CMU-CS-82-119, Carnegie Mellon University Computer Science Department, May, 1982.

This paper describes systolic algorithms for a number of geometric problems. Implementations yielding maximal throughput are given for solving dynamic versions of convex hull, inclusion, range and inverse range search, point location, intersection, triangulation, and closest-point problems.

- [Dohi 82] Dohi, Y., A.L. Fisher, H.T. Kung and L.M. Monier.
The Programmable Systolic Chip: Project Overview.
In *Proceedings of Workshop on Algorithmically-Specialized Computer Organizations*, September, 1982.

- [Ebeling 83] Ebeling, C. and O. Zajicek.
Validating VLSI Circuit Layout by Wirelist Comparison.
In *Proceedings of the 10th Annual International Symposium on Computer Architecture*, Pages 172-173. IEEE, September, 1983.
The correctness of a VLSI circuit layout can be accomplished by comparing the wirelist extracted from the layout with the specification wirelist. We describe a program that compares wirelists using a very efficient graph isomorphism algorithm that works well for practical circuits.

- [Ebeling 84a] Ebeling, C.E.
A VLSI Chess Move Generator.
In *Proceedings of the 11th Annual International Symposium on Computer Architecture*, June, 1984.

- [Ebeling 84b] Ebeling, C.
A Minimal Software Package for the CMU Chip Tester.
Technical Report VLSI Memo #149, Carnegie Mellon University Computer Science Department, 1984.

- [Fisher 81] Fisher, A.
Systolic Algorithms for Running Order Statistics in Signal and Image Processing.
In Kung, H.T., R.F. Sproull, and G.L. Steele Jr., *VLSI Systems and Computations*, Pages 265-272. Computer Science Press, Inc., 1981.
Median smoothing, a filtering technique with wide application in digital signal and image processing, involves replacing each sample in a grid with the median of the samples within some local neighborhood. As implemented on conventional computers, this operation is extremely expensive in both computation and communication resources. This paper defines the running order statistics (ROS) problem, a generalization of median smoothing. It then summarizes some of the issues involved in the design of special purpose devices implemented with very large scale integration (VLSI) technology. Finally, it presents algorithms designed for VLSI implementation which solve the ROS problem and are efficient with respect to hardware resources, computation time, and communication bandwidth.

- [Fisher 82a] Fisher, A.L.
Systolic Algorithms for Running Order Statistics in Signal and Image Processing.
Journal of Digital Systems VI(2/3):251-264, Summer/Fall, 1982.
A preliminary version appears in Kung, H.T., R.F. Sproull, and G.L. Steele, Jr. (editors), *VLSI Systems and Computations*, Computer Science Press, Inc., 1981.

- [Fisher 82b] Fisher, A.L. and H.T. Kung.
Synchronize Large Systolic Arrays.
In *Proceedings of the SPIE, Vol. 341, Real-Time Signal Proceedings V*, SPIE, May, 1982.
- [Fisher 82c] Fisher, A.L. and H.T. Kung.
Special-Purpose VLSI Architectures: General Discussions and a Case Study.
In *Proceedings of the USC Workshop on VLSI and Modern Signal Processing*, November, 1982.
Cited in Kung and Lin 83 'An Algebra for VLSI Algorithm Design'.
- [Fisher 83a] Fisher, A.L. and H.T. Kung.
Synchronizing Large VLSI Processor Arrays.
In *Proceedings of the 10th Annual International Symposium on Computer Architecture*, Pages 54-58. Jun, 1983.
Highly parallel VLSI computing structures consist of many processing elements operating simultaneously. In order for such processing elements to communicate among themselves, some provision must be made for synchronization of data transfer. The simplest means of synchronization is the use of a global clock. Unfortunately, large clocked systems can be difficult to implement because of the inevitable problem of clock skews and delays, which can be especially acute in VLSI systems as features sizes shrink. For the near term, good engineering and technology improvements can be expected to maintain the feasibility of clocking in such systems; however, clock distribution problems crop up in any technology as systems grow. An alternative means of enforcing necessary synchronization is the use of self-timed, asynchronous schemes, at the cost of increased design complexity and hardware cost. Realizing that different circumstances call for different synchronization methods, this paper provides a spectrum of synchronization models; based on the assumptions made for each model, theoretical lower bounds on clock skew are derived, and appropriate or best-possible synchronization schemes for large processor arrays are proposed.
One set of models is based on assumptions that allow the use of pipelined clocking scheme, where more than one clock event is propagated at a time.
- [Fisher 83b] Fisher, A.L., H.T. Kung, L. M. Monier, H. Walker and Y. Dohi.
Design of the PSC: A Programmable Systolic Chip.
In Bryant, R., Editor, *Proceedings of the Third Caltech Conference on Very Large Scale Integration*, Pages 287-302. California Institute of Technology, March, 1983.
The CMU *programmable systolic chip* (PSC) is a high performance, special-purpose, single-chip microprocessor intended to be used in groups of tens or hundreds for the efficient implementation of a broad variety of systolic arrays. For implementing these systolic arrays, the PSC is expected to be at least an order of magnitude more efficient than conventional microprocessors. The development of the PSC design, from initial concept to a silicon layout, took slightly less than a year. The PSC project represents an integration of many disciplines including applications, algorithms, architecture, microprocessor design, and chip layout. This paper describes the goals of the project, the design process, major design features and current status.

- [Fisher 83c] Fisher, A.L., H.T. Kung, L.M. Monier, and Y. Dohi.
Architecture of the PSC: A Programmable Systolic Chip.
In *Proceedings of the 10th Annual International Symposium on Computer Architecture*, Pages 48-53. IEEE, June, 1983.
A later version appears in *Journal of VLSI and Computer Systems*. See Fisher 84b for abstract.
- [Fisher 84a] Fisher, A.L.
Dictionary Machines with a Small Number of Processors.
In *Proceedings of the 11th Annual International Symposium on Computer Architecture*, IEEE, June, 1984.
A number of tree-structured multiprocessor designs have been proposed for performing a group of dictionary operations (INSERT, DELETE, EXTRACTMIN, NEAR, etc.) on a set of keys. These designs typically use one processor for each key stored and operate with constant throughput, assuming unit time to communicate and compare keys. This assumption breaks down in applications with long keys. This paper describes a machine which uses a number of processors proportional to the maximum length of a key to achieve constant throughput, regardless of key length. This design has important practical advantages over the family of tree-structured machines, and demonstrates that processor-intensive VLSI structures are not always the best route to a high-performance system.
- [Fisher 84b] Fisher, A.L., H.T. Kung, L.M. Monier and Y. Dohi.
The Architecture of a Programmable Systolic Chip.
*Journal of VLSI and Computer Systems*1(2):153-169, 1984.
An earlier version appears in *Conference Proceedings of the 10th Annual Symposium on Computer Architecture*, Stockholm, Sweden, June 1983, pp. 48-53.
In recent years, many systolic algorithms have been proposed as solutions to computationally demanding problems in signal and image processing and other areas. Such algorithms exploit the regularity and parallelism of problems to achieve high performance and low I/O requirements. Since systolic algorithms generally consist of a few types of simple processors, or systolic cells, connected in a regular pattern, they are less expensive to design and implement than more general machines.
This advantage is offset by the fact that a particular system can generally be used only on a narrow set of problems, and thus design cost cannot be amortized over a large number of units. One way to approach this problem is to provide a programmable systolic chip (PSC), many copies of which can be connected and programmed to implement many systolic algorithms.
The systolic environment, by virtue of its emphasis on continuous, regular flow of data and fairly simple pre-cell processing, imposes new design requirements for programmable processors which are quite different from those found in a general-purpose system. This paper describes the CMU PSC, a single-chip microprocessor suitable for use in groups of tens or hundreds for the efficient implementation of a broad variety of systolic arrays. The processor has been fabricated in nMOS, and is undergoing testing.
- [Fisher 84c] Fisher, A.L., H.T. Kung and, K. Sarocky.
Experience with the CMU Programmable Systolic Chip.
In *Proceedings of SPIE Symposium, Vol. 495, Real-Time Signal Processing VII*, SPIE, August, 1984.
The CMU programmable systolic chip (PSC) is an experimental, microprogrammable

chip designed for the efficient implementation of a variety of systolic arrays. The PSC has been designed, fabricated, and tested. The chip has about 25,000 transistors, uses 74 pins, and was fabricated through MOSIS, the DARPA silicon broker, using a 4 micron nMOS process. A modest demonstration system involving nine PSCs is currently running. Larger demonstrations are ready to be brought up when additional working chips are acquired.

The development of the PSC, from initial concept to a silicon layout, took slightly less than a year, but testing, fabrication, and system demonstration took an additional year. This paper reviews the PSC, describes the PSC demonstration system, and discusses some of the lessons learned from the PSC project.

- [Forgy 84] Forgy, C., A. Gupta, A. Newell, R. Wedig.
Initial Assessment of Architectures for Production Systems.
In *AAAI-84, National Conference for Artificial Intelligence*, AAAI, August, 1984.
- [Foster 81] Foster, M.J.
Syntax-Directed Verification of Circuit Function.
In *Conference on VLSI Systems and Computations*, Pages 196-202. October, 1981.
- [Foster 82] Foster, M.J. and H.T. Kung.
Recognize Regular Languages With Programmable Building Blocks.
Journal of Digital Systems 6(4):323-332, 1982.
Also available as CMU-CSD technical report CMU-CS-81-126.
This paper introduces a new programmable building-block for recognition of regular languages. By combining three types of basic cells a circuit for recognizing any regular language can be constructed or "programmed" automatically from the regular expression describing that language. Recognizers built in this way are efficient pipeline circuits that have constant response time and avoid broadcast. In addition, the paper proposes the use of a single, regular layout, called the PRA (programmable recognizer array), that can be "personalized" to recognize the language specified by any regular expression. PRA's provide compact reconfigurable layouts for recognizer circuits, requiring only $O(n \log n)$ area for regular expressions of length n .
- [Foster 83] Foster, M.J.
A Laser-Programmable Chip for Language Recognition.
In *Proceedings of the IEEE Workshop on Languages for Automation*, IEEE, November, 1983.
A similar paper appeared in '84 MIT VLSI conference. See Foster, 'E.T.: A Laser-Programmed Language Recognizer Chip' Jan. 1984 reference for abstract.
- [Foster 84] Foster, M.J.
E.T.: A Laser-Programmed Language Recognizer Chip.
In Paul Penfield, Editor, *Proceedings, Conference on Advanced Research in VLSI*, Pages 171-178. MIT, January, 1984.
A similar paper appeared in *Proceedings of the IEEE Workshop on Languages for Automation*, Nov. 1983.
Programmable layouts for specialized application areas allow the rapid design of efficient custom chips. This paper presents a prototype programmable layout for language recognizers. The prototype chip was fabricated in NMOS, and programmed by cutting aluminum lines using a laser. Design considerations, pro-

gramming expertise, and test results for this chip are presented, and guidelines for the design of future chips are extracted from this experience.

- [Frank 81a] Frank, M.J. and H.T. Kung.
Two Timing Samplers.
In *Proceedings of the 1981 CalTech VLSI Conference*, CalTech, February, 1981.
- [Frank 81b] Frank, E.H. and R.F. Sproull.
Testing and Debugging Custom Integrated Circuits.
Technical Report CMU-CS-81-105, Carnegie Mellon University Computer Science Department,
February, 1981.
Also in *Computing Surveys* (13)4, pp.425-451, December, 1981.
- [Frank 81c] Frank, E.H., Ebeling, C.E., and Sproull, R.F.
Hierarchical Wirelist Format.
Technical Report VLSI Document V087, Carnegie Mellon University Computer Science Department,
1981.
- [Frank 82] Frank, E.
The Fast-1: A Data-Driven Multiprocessor for Logic Simulation.
Technical Report VLSI Document V121, Carnegie Mellon University Computer Science Department,
October, 1982.
- [Frank 83] Frank, E.D. and R.F. Sproull.
A Self-Timed Static RAM.
In Bryant, R., Editor, *Proceedings of the Third Caltech Conference on Very Large Scale Integration*, Pages 275-285. California Institute of Technology, March, 1983.
This paper presents the design of a self-timed static RAM. Although the memory array uses a conventional six-transistor static cell, extra circuitry is associated with each column and each row of the memory to make the memory self-timed. This circuitry detects several conditions: address line precharge complete, word line driven, bit line precharge complete, read complete, and write complete. To increase the read speed, each column of the memory uses an unlocked sense amplifier. The memory design includes a controller that implements a four-phase request/acknowledge interface. Although the memory is intended for use as part of a single-chip processor and not as a stand-alone chip, we have fabricated a 64 by 64 bit test chip and measured its performance.
- [Gentleman 81] Gentleman, W.M. and H.T. Kung.
Matrix Triangularization by Systolic Arrays.
In *Proceedings of SPIE Symposium, Vol. 298, Real-Time Processing IV*, Pages 19-26. Society of Photo-Optical Instrumentation Engineers, August, 1981.
- [Gupta 81a] Gupta, S. and R.F. Sproull.
Filtering Edges for Gray-Scale Displays.
Computer Graphics 15(3), August, 1981.

- [Gupta 81b] Gupta, S., R.F. Sproull, and I.F. Sutherland.
A VLSI Architecture for Updating Raster-Scan Displays.
Computer Graphics 15(3), August, 1981.
- [Gupta 83a] Gupta, A.
ACE: A Circuit Extractor.
In *Proceedings of the 20th Design Automation Conference*, IEEE, June, 1983.
This paper describes the design, implementation and performance of a flat edge-based circuit extractor for NMOS circuits. The extractor is able to work on large and complex designs, it can handle geometry, and outputs a comprehensive wirelist. Measurements show that the run time of the edge-based algorithm used is linear in size of the circuit, with low implementation overheads. The extractor is capable of analyzing a circuit with 20,000 transistors in less than 30 minutes of CPU time on a VAX 11/780. The high performance of the extractor has changed the role that a circuit extractor played in the design process, as it is now possible to extract a chip a number of times during the same session.
- [Gupta 83b] Gupta, A. and R.W. Hon.
HEXT: A Hierarchical Circuit Extractor.
Journal of VLSI and Computer Systems, Spring, 1983.
This paper describes the algorithms, implementation, and performance of a hierarchical circuit extractor for NMOS designs. The input to the circuit extractor is a description of the layout of the chip, and its output is a hierarchical wirelist describing the circuit. The extractor is divided into two parts, a front-end and a back-end. The front-end analyzes the CIF description of a layout and partitions it into a set of non-overlapping rectangular regions called *windows*; redundant windows are recognized and are extracted only once. The back-end analyzes each unique window found by the front-end. The back-end determines the electrical circuit represented by the window, and computes an interface that is later used to combine the window with others that are adjacent. The paper also presents a simple analysis of the expected performance of the algorithm, and the results of running the extractor on some real chip designs.
- [Gupta 84a] Gupta, A.
Parallelism in Production Systems: The Sources and the Expected Speed-up.
Technical Report CMU-CS-84-169, Carnegie Mellon University Computer Science Department,
December, 1984.
Production systems (or rule-based systems) are widely used in Artificial Intelligence for modeling intelligent behavior and building expert systems. On the surface production systems appear to be capable of using large amounts of parallelism—it is possible to perform match for each production in parallel. Initial measurements and simulations, however, show that the speed-up available from such use of parallelism, is quite small. The limited speed-up available from the obvious sources has led us to explore other sources of parallelism. This paper represents an initial attempt to identify the various sources of parallelism in production system programs and to characterize them, that is, to determine the potential speed-up offered by each source and the overheads associated with it. The paper also addresses some implementation issues related to using the various sources of parallelism.

[Gupta 84b]

Gupta, A.

Implementing OPS5 Production Systems on DADO.

In *International Conference on Parallel Processing*, 1984.

DADO is a highly parallel tree-structured architecture designed to execute *production systems* at Columbia University. In this paper, we analyze the performance of DADO when executing OPS5 production system programs. The analysis is based on the predicted performance of three different algorithms for implementing production systems on DADO. Our analysis shows that the large-scale parallelism in DADO is not very effective for executing OPS5-like production systems. The reasons are: (1) actions of productions in OPS5 programs do not have global affects, but only affect a small number of other productions, and (2) large-scale parallelism almost always implies that the individual processing elements are weak. Since only a small number of productions are affected every cycle, only a few of the large number of processing elements perform useful work. Furthermore, since the individual processing elements are weak, the performance is worse than if a small number of powerful processors are used. The tree-structured topology of the DADO architecture is not found to be a bottleneck.

[Haynes 82]

Haynes, L.S., R.L. Lau, D.P. Siewiorek and D. W. Mizell.

A Survey of Highly Parallel Computing.

Computer 15(1), January, 1982.

[Hon 83]

Hon, R.W.

The Hierarchical Analysis of VLSI Designs.

Technical Report CMU-CS-83-170, Carnegie Mellon University Computer Science Department,

December, 1983.

As the complexity of integrated circuit designs increases, the task of verifying that the masks are correct becomes very time consuming. Fortunately, the wide acceptance of hierarchical mask description formats allows the development of methods that take advantage of the structure in such descriptions. These hierarchical processing methods are potentially much faster (for actual designs) than methods that ignore hierarchy.

This thesis explores one method of hierarchically processing integrated circuit artwork. The method has the following properties:

1. The same method is applicable to a range of artwork analysis tasks, for example plotting, circuit extraction, and design-rule checking.
2. The method operates on mask descriptions, and can directly replace existing slower techniques.
3. There are no restrictions on the type of designs that can be analyzed, so integrated circuits created using any number of design systems can be checked.

Informal arguments for the method's correctness, performance results, and an overview of where this type of design aid fits in the spectrum of VLSI tools are given.

[Hong 81]

Hong, J. W. and H.T. Kung.

I/O Complexity: The Red-Blue Pebble Game.

In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, Pages 326-333. ACM SIGACT, May, 1981.

Also available as CMU-CSD technical report CMU-CS-81-120.

In this paper, the *red-blue pebble game* is proposed to model the input-output complexity of algorithms. Using the pebble game formulation, a number of lower

bound results for the I/O requirement are proven. For example, it is shown that to perform the n -point FFT (or the ordinary $n \times n$ matrix multiplication algorithm) with a device of $O(S)$ memory, at least $\Omega(n \log n / \log S)$ (or $\Omega(n^2 / \sqrt{S})$, respectively) time is needed for the I/O. Similar results are obtained for algorithms for several other problems. All of the lower bounds presented are the best possible in the sense that they are achievable by certain decomposition schemes.

The results in this paper provide insight into the difficult task of balancing I/O and computation in special-purpose system design. For example, for the n -point FFT, the I/O lower bound implies that an S -point device achieving a speed-up ratio $O(\log S)$ over the conventional $O(n \log n)$ implementation is all that one can hope for.

[Hsu 84]

Hsu, F.H., H.T. Kung, T. Nishizawa, and A. Sussman.

LINC: The Link and Interconnection Chip.

Technical Report CMU-CS-84-159, Carnegie Mellon University Computer Science Department, May, 1984.

The link and interconnection chip (LINC) is a custom chip whose function is to serve as an efficient link between system functional modules, such as arithmetic units, register files, and I/O ports.

LINC has 4-bit datapaths consisting of an 8×8 crossbar interconnection, a FIFO or programmable delay for each of its inputs, and a pipeline register file for each of its outputs. Using pre-stored control patterns, LINC can configure its interconnection and delays on-the-fly, while running. Therefore the usual functions of busses and register files can be realized with this single chip.

LINC can be used in a bit-sliced fashion to form interconnections with datapaths wider than 4 bits. Moreover, by tri-stating the proper data output pins, multiple copies of LINC can form crossbar interconnections larger than 8×8 .

Operating at the target cycle time of 100 ns, LINC makes it possible to implement a variety of high-performance processing elements with much reduced package counts. This reduction of chip counts is especially significant for cost-effective implementations of those multiprocessors such as systolic arrays which call for large numbers of processing elements.

This paper gives the architectural specification of LINC, and justifies the specification by some application examples.

[Kung 81a]

Kung, H.T., L.M. Ruane, and D.W.L. Yen.

A Two-Level Pipelined Systolic Array for Convolutions,

In Kung, H.T., G.L. Steele, Jr., and R.F. Sproull, *VLSI Systems and Computations*. Computer Science Press, Inc., 1981.

Pipelining computations over a large array of cells has been an important feature of systolic arrays. To achieve even higher degrees of concurrency, it is desirable to have cells of a systolic array themselves be pipelined as well. The resulting two-level pipelined systolic array would enjoy in principle a k -fold increase in its throughput, where k is the ratio of the time to perform the entire cell computation over that to perform just one of its pipeline stages. This paper describes such a two-level pipelined systolic array that is capable of performing convolutions of any dimension. The designs take full advantages of the pipelining assumed to be available at each cell.

Multi-stage pipelined arithmetic units built from discrete components have been used

in most of high-performance computers. With the advent of VLSI, these pipelined units will surely be implemented in one or few chips. This paper shows for the first time how a large number of these pipelined chips can be efficiently combined to form a systolic array.

- [Kung 81b] Kung, H.T.
Design of Algorithms for Direct VLSI Implementation,
In Cavalli, E., *Design of Numerical Algorithms for Parallel Processing*, Academic Press, 1981.
- [Kung 81c] Kung, H.T., G.L. Steele Jr., and R.F. Sproull (eds.).
VLSI Systems and Computations,
Computer Science Press, Inc., Maryland, 1981.
- [Kung 81d] Kung, H.T.
Use of VLSI in Algebraic Computation: Some Suggestions.
In Wang, P.S., Editor, *Proceedings of 1981 ACM Symposium on Symbolic and Algebraic Computation*, Pages 218-222. ACM SIGSAM, August, 1981.
- [Kung 81e] Kung, H.T. and R.L. Picard.
Hardware Pipelines for Multi-Dimensional Convolution and Resampling.
In *Proceedings of the 1981 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Pages 237-278. IEEE, November, 1981.
A later version appears as 'One-Dimensional Systolic Arrays for Multidimensional Convolution and Resampling,' 1984.
- [Kung 82a] Kung, H.T. and S.W. Song.
A Systolic 2-D Convolution Chip,
In Preston, K., Jr. and L. Uhr, *Multicomputers and Image Processing: Algorithms and Programs*, Pages 373-384. Academic Press, 1982.
Also available as CMU-CSD technical report CMU-CS-81-110.
This paper describes a chip for performing the 2-D (two-dimensional) convolution in signal and image processing. The chip, based on a systolic design, consists of essentially only one type of simple cells, which are mesh-interconnected in a regular and modular way, and achieves high performance through extensive concurrent and pipelined use of these cells. Denoting by u the cycle time of the basic cell, the chip allows convolving a $k \times k$ window with an $n \times n$ image in $O(n^2 u / k)$ time, using a total of k^3 basic cells. The total number of cells is optimal in the sense that the usual sequential algorithm takes $O(n^2 k^2 u)$ time. Furthermore, because of the modularity of the design, the number of cells used by the chip can be easily adjusted to achieve any desirable balance between I/O and computation speeds.
- [Kung 82b] Kung, H.T. and S.Q. Yu.
Integrating High-Performance Special-Purpose Devices into a System.
In *Notes for Symposium on Vector and Parallel Processors*, IBM Italy Scientific Center, March, 1982.
See Kung and Yu, May 1982, SPIE Vol. 341 Real Time Signal Processing V, for abstract.
- [Kung 82c] Kung, H.T.
Why Systolic Architectures?
Computer Magazine 15(1):37-46, January, 1982.

[Kung 82d]

Kung, H.T. and S.Q. Yu.

Integrating High-Performance Special-Purpose Devices into a System.

In *SPIE Vol. 341 Real Time Signal Processing V (1982)*, SPIE, May, 1982.

An emerging belief among many researchers is that a significant portion of the next generation of high performance computers will be based on architectures capable of exploiting very large scale integration (VLSI) modules. In particular, it is desirable to have a compact system that can be plugged in with interchangeable high performance modules to fit various application requirements. The system can be an efficient signal processor when special purpose signal processing modules are used; it can also be an efficient database machine when the modules are replaced with data processing modules. This paper discusses some of the issues in the design of such a system, and describes the framework of a system that is being developed at CMU.

[Kung 83a]

Kung, H.T., L.M. Ruane, and D.W.L. Yen.

Two-Level Pipelined Systolic Array for Multidimensional Convolution.

*Image and Vision Computing*1(1):30-36, February, 1983.

Also available as CMU-CSD technical report CMU-CS-83-103.

This paper describes a systolic array for the computation of n -dimensional (nD) convolutions for any positive integer n . Systolic systems usually achieve high performance by allowing computations to be pipelined over a large array of processing elements. To achieve even higher performance, the systolic array described in this paper uses a second level of pipelining by allowing the processing elements themselves to be pipelined to an arbitrary degree.

[Kung 83b]

Kung, H.T.

Two-Level Pipelined Systolic Arrays for Matrix Manipulation, Polynomial Evaluation, and Discrete Fourier Transform.

In *Proceeding of the Workshop on Dynamical Behavior of Automata: Theory and Applications*, Academic Press, September, 1983.

In recent years many systolic algorithms have been designed and several prototypes of systolic array processors have been constructed. Major efforts have now started in attempting to use systolic array processors in large, real-life applications. Practical issues on the implementation of systolic array processors have begun to receive substantial attention.

One of the important implementation issues relates to the efficient use of pipelined functional units in the implementation of systolic cells. For example, high throughput floating-point multiplier and adder circuits typically employ three or more pipeline stages. Systolic cells implemented using these units form a *second level of pipelining* in the pipelined organization of systolic arrays. This additional level of pipelining can greatly increase the system throughput.

[Kung 83c]

Kung, H.T. and W.T. Lin.

An Algebra for VLSI Algorithm Design.

Technical Report CMU-CS-84-100, Carnegie Mellon University Computer Science Department,

April, 1983.

Algorithms designed for VLSI implementation are usually parallel and two-dimensional in the sense that many processing elements laid out on a silicon surface can operate simultaneously. These algorithms have been typically described by graphs or networks where nodes represent processing elements or registers and

edges represent wires. Although for many purposes these traditional representations are adequate for specifying VLSI algorithms, they are not suited for manipulating algorithm designs. In this paper an algebraic representation, together with a semantics, is proposed for VLSI algorithm designs. By algebraic transformations analogous to some typically used in linear algebra, alternative but equivalent designs satisfying desirable properties such as locality and regularity in data communication can be derived. This paper describes this powerful algebra for manipulating designs, and provides a mathematical foundation for the algebraic transformations. The algebraic framework is more suitable for supporting formal manipulation of designs than the network or graph-theory models, especially for complex designs. As an application of the proposed algebra, the paper demonstrates its use in the design and verification of systolic algorithms.

[Kung 83d]

Kung, H.T. and M. Lam.

Fault-Tolerance and Two-Level Pipelining in VLSI Systolic Arrays.

Technical Report CMU-CS-83-166, Carnegie Mellon University Computer Science Department,

November, 1983.

This paper addresses two important issues in systolic designs: fault-tolerance and two-level pipelining. The proposed 'systolic' fault-tolerant scheme maintains the original data flow pattern by bypassing defective cells with a few registers. As a result, many of the desirable properties of systolic arrays (such as local and regular communication between cells) are preserved. Two-level pipelining uses pipelined units to increase the overall system. We show that both of these problems can be reduced to the same mathematical problem of incorporating extra delays on certain data paths in originally correct systolic designs. We introduce the mathematical notion of a *cut* which enables us to handle this problem effectively.

The results obtained are encouraging. When delays are added to systolic arrays without feedback cycles, the arrays can tolerate large numbers of failures (with the addition of very little hardware) while maintaining the original throughput. However, adding delays to systolic arrays with cycles typically induces a significant decrease in throughput. In response to this, we have derived a new class of systolic algorithms to cycle the data around a ring of processing cells. The *systolic ring* architecture's performance degrades gracefully as cells fail.

[Kung 83e]

Kung, H.T.

A High Performance Microprocessor Chip to Be Used in Groups of Hundreds.

In *Proceedings of IEEE EASCON '83*, Pages 251-258. IEEE, September, 1983.

[Kung 83f]

Kung, H.T. and W.T. Lin.

An Algebra for VLSI Computation,

In Birkhoff, G. and A. Schoenstadt, *Elliptic Problem Solvers II*. Academic Press, Orlando, 1983.

[Kung 83g]

Kung, H.T.

VLSI, Computer Science, and Synergetic Research.

In *Proceedings of the ACM 11th Annual Computer Science Conference*, Pages 17-19. February, 1983.

- [Kung 83h] Kung, H.T.
On the Implementation and Use of Systolic Array Processors.
In *Proceedings of International Conference on Computer Design: VLSI in Computers*, Pages 370-373. IEEE, November, 1983.
In recent years many systolic algorithms have been designed and several prototypes of systolic array hardware have been constructed. Major efforts now started in attempting to use systolic array processors in large, real-life applications; practical issues on the implementation and use of systolic array processors in systems have begun to receive substantial attention. This paper first examines various implementation issues and alternatives, and then identifies some work that is essential to the eventual, wide use of systolic array processors.
- [Kung 83i] Kung, H.T. and S.Q. Yu.
Integrating High-Performance Special-Purpose Devices into a System,
In Randel, B. and Treleaven, P.C., *VLSI Architecture*, Pages 205-211. Prentice/Hall International, 1983.
An emerging belief among many researchers is that a significant portion of the next generation of high performance computers will be based on architectures capable of exploiting very large scale integration (VLSI) modules. In particular, it is desirable to have a compact system that can be plugged in with interchangeable high performance modules to fit various application requirements. The system can be an efficient signal processor when special purpose signal processing modules are used; it can also be an efficient database machine when the modules are replaced with data processing modules. This paper discusses some of the issues in the design of such a system, and describes the framework of a system that is being developed at CMU.
- [Kung 84a] Kung, H.T. and R.L. Picard.
One-Dimensional Systolic Arrays for Multidimensional Convolution and Resampling,
In Fu, King-sun, *VLSI for Pattern Recognition and Image Processing*, Pages 9-24. Springer-Verlag, 1984.
We present one-dimensional systolic arrays for performing two- or higher-dimensional convolution and resampling. These one-dimensional arrays are characterized by the fact that their I/O bandwidth requirement is independent of the size of the convolution kernel. This contrasts with alternate two-dimensional array solutions, for which the I/O bandwidth must increase as the kernel size increases. The proposed architecture is ideal for VLSI implementation - and arbitrarily large kernel can be handled by simply extending the linear systolic array with simple processors of the same type, so that one processor corresponds to each kernel element.
- [Kung 84b] Kung, H.T. and W.T. Lin.
An Algebra for Systolic Computation,
In Birkhoff, G. and A. Schoenstadt, *Elliptic Problem Solvers II*, Pages 141-160. Academic Press, 1984.

- [Kung 84c] Kung, H.T. and M.S. Lam.
Fault-Tolerance and Two-Level Pipelining in VLSI Systolic Arrays.
In *Proceedings of Conference on Advanced Research in VLSI*, MIT, January, 1984.
Also available as CMU-CSD technical report CMU-CS-83-166. A revised version appears
in *Journal of Parallel and Distributed Computing*, vol. 1, 1984. For abstract, see Kung 84
"Wafer-Scale Integration and Two-Level Pipelined Implementation of Systolic Arrays."

- [Kung 84d] Kung, H.T.
Systolic Algorithms and Their Implementation.
In *Proceedings of the 17th Hawaii International Conference on System Sciences*, Pages 5-11.
January, 1984.

- [Kung 84e] Kung, H.T. and M. Lam.
Wafer-Scale Integration and Two-Level Pipelined Implementation of Systolic Arrays.
Journal of Parallel and Distributed Computing 1:32-63, 1984.
A preliminary version appeared in *Proceedings of the Conference on Advanced Research in VLSI*, MIT, January 1984.

Two important issues in systolic array designs are addressed: How is fault tolerance provided in systolic arrays to enhance the yield of wafer-scale integration implementations? And, how are efficient systolic arrays with two levels of pipelining designed? (The first level refers to the pipelined organization of the array at the cellular level, and the second refers to the pipelined functional units inside the cells.) The fault-tolerant scheme proposed replaces defective cells with clocked delays. The mathematical notion of a *cut* is introduced to solve the problem of how to allow for the extra delays in the data paths while preserving the correctness of the original systolic array designs.

The results obtained by applying these techniques are encouraging. When applied to systolic arrays without feedback cycles, the arrays can tolerate large numbers of failures while maintaining the original throughput. Furthermore, by adding a small number of delay registers, all the pipeline stages in the cells can be kept fully utilized. However, adding delays to systolic arrays with cycles typically induces a significant decrease in throughput.

In response to this, a new class of systolic algorithms has been derived in which the data cycle around a ring of processing cells. The *systolic ring* architecture has the property that its performance degrades gracefully as cells fail. Use of the cut theory and ring architectures for arrays with feedback gives effective fault-tolerant and two-level pipelining schemes for most systolic arrays. As a side effect of developing the ring architecture approach, several new systolic algorithms have been derived.

- [Lam 83] Lam, M. and J. Mostow.
A Transformational Model of VLSI Systolic Design.
In *Proceedings of the 6th International Symposium on Computer Hardware Description Languages and their Applications*, Pages 65-77. IFIP, May, 1983.
A later version appears in *Computer*, Feb. 1985.

This paper presents a transformational model and convenient notation for systolic design. The model is implemented in a program that accepts a software algorithm, along with a bit of advice, and applies a series of transformations to produce a functional-level circuit description. The simplicity of the program and the clarity of the notation appear largely due to two factors:

1. The representation of a design is factored into a *structure* description, which

specifies the hardware components and their interconnections, and a *driver*, which relates the input and output data streams for the structure to the variables used in the algorithm.

2. The notation includes constructs that make it easy to represent timing and communication schemes common in systolic design.

- [Lehman 81] Lehman, P.L.
A Systolic (VLSI) Array for Processing Simple Relational Queries.
In *VLSI Systems and Computations*, Pages 285-295. Computer Science Press, Inc., Oct, 1981.
This paper discusses the use of *systolic arrays* (a conceptual and design tool for VLSI systems) to produce VLSI capable of processing simple relational database queries, which are *by far* the most frequently executed queries in practical large database systems. We will be concerned with the exploitation of VLSI technology to process "simple" relational queries *very rapidly*; the design of an array for this task is described below. The systolic properties of the array design are considered, and are shown to have analogs in the domain of databases by using the systolic properties to prove certain consistency and scheduling complexity properties of all transactions executed by the array (hereinafter called the *simple query array*, or SQA). The SQA is intended for use as an integral part of a *systolic database machine*, which would handle very large databases and is expected to have a high performance gain over conventional database systems. The machine should also compare quite favorably with other database machine designs, especially when use for databases with frequent simple queries, *i.e.* those databases used by most commercial applications.
- [Leiserson 81] Leiserson, C.E. and J.B. Saxe.
Optimizing Synchronous Systems.
In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*, Pages 23-36. IEEE Computer Society, October, 1981.
- [Leiserson 83a] Leiserson, C.E.
Area-Efficient VLSI Computation.
PhD thesis, Carnegie-Mellon University, 1983.
- [Leiserson 83b] Leiserson, C.E. and J.B. Saxe.
Optimizing Synchronous Systems.
*Journal of VLSI and Computer Systems*1(1):41-68, 1983.
- [Luk 84a] Luk, W.K.
ROUTER: A Set of Routing Tools for VLSI Layout.
Technical Report VLSI Document V155, Carnegie Mellon University Computer Science Department,
April, 1984.
- [Luk 84b] Luk, W.T.
A Greedy Switch-box Router.
Technical Report CMU-CS-84-148, Carnegie Mellon University Computer Science Department,
May, 1984.
We show how to extend the greedy channel router of Rivest and Fiduccia into an

efficient switch-box router. Terminals are on the boundary of a rectangular region, and the router uses two orthogonal layers of wires to generate the solution. The router always succeeds in finding a solution by inserting sufficient horizontal and vertical tracks in case of failure. The result is generated through a single scan of the routing region. The implemented router is designed for assembling custom VLSI design, it works in parallel with other tools such as a layout editor which serves as an interface. The router output is in CIF.

- [Luk 84c] Luk, W.K. and J.E. Vuillemin.
Recursive Implementation of Optimal Time VLSI Integer Multipliers,
Advances in Computing Research. JAI Press, Inc., 1984.
- [Oflazer 83] Oflazer, K.
Design and Implementation of a Single-Chip 1-D Median Filter.
In *IEEE Transactions on Acoustics, Speech, and Signal Processing*, IEEE, October, 1983.
Also available as CMU-CSD techreport CMU-CS-82-115.
The design and implementation of a VLSI chip for the one-dimensional median filtering operation is presented. The device is designed to operate on 8-bit sample sequences with a window size of five samples. Extensive pipelining and employment of systolic data-flow concepts at the bit level enable the chip to filter at rates up to ten megasamples per second. A configuration for using the chip for approximate two-dimensional median filtering is also presented.
- [Oflazer 84] Oflazer, K.
Partitioning in Parallel Processing of Production Systems.
In Keller, R., Editor, *Proceedings of the International Conference on Parallel Processing*, ACM, IEEE, and Department of Computer and Information Science, Ohio State University at Columbus, August, 1984.
The results of an analysis of production level parallelism in OPS5 production system programs is presented. The results indicate that contrary to most expectations, the effective production level parallelism in this class of production systems considered is very low compared to the number of productions in these systems. Hence, significant speed-ups in executing such systems would be obtained by combining the limited parallelism with fast hardware and overlapped processing; rather than by massively parallel approaches employing simple processors. Later, the problem of partitioning productions in a production system to a small number of processors in a parallel processing system is presented. The goal of partitioning is to improve the speed-up provided by the limited parallelism by finding assignments of productions to processors that achieve a more balanced load for each processor.
- [Ostlund 82] Ostlund, N., P. Hibbard, and R. Whiteside.
A Case Study in the Application of a Tightly Coupled Multiprocessor to Scientific Computations,
In Alder, B., S. Fernbach, and M. Rotenberg, *Parallel Computations*. Academic Press, Inc., 1982.
- [Raibert 82a] Raibert, M.H. and J.E. Tanner.
A VLSI Tactile Array Sensor.
In *Proceedings International Symposium on Industrial Robots*, 1982.
A new type of tactile sensor is presented that was designed to give a robot manipulation system information about contact between its hand and objects in the environ-

ment. We describe a device that is at once a special purpose parallel computer and a high resolution tactile sensing array. We have replaced the passive substrate of earlier tactile sensors with a custom designed nMOS VLSI device that handles transduction, computing and communication. Forces are transduced using a standard conductive plastic technique. An array of processors, the sensor performs filtering and simple convolution operations on the tactile image. Data are then read from the array serially and transmitted to a control computer. A 6x3 array with 1 mm square tactile cells has been fabricated and is working in the laboratory. Larger devices, up to 30 x 30 cells, are currently being designed.

[Raibert 82b]

Raibert, M.H. and J.E. Tanner.

Design and Implementation of a VLSI Tactile Sensing Computer.

*Robotics Research*1(3), Fall, 1982.

A new type of tactile sensor is presented that was designed to give a robot manipulation system information about contact between its hand and objects in the environment. We describe a device that is at once a special purpose parallel computer and a high resolution tactile array sensor. The passive substrates of earlier tactile sensors have been replaced with a custom-designed very large scale integration (VLSI) device that performs transduction, tactile image processing, and communication. Forces are transduced using a conductive plastic technique in conjunction with metal electrodes on the surface of an integrated circuit. An array of processors implemented within the integrated circuit perform parallel two-dimensional convolutions between programmable filtering masks and a binary tactile image. Data are then read from the array serially, so they can be transmitted to a control computer. A 6x3 array sensor with 1 mm² tactile cells has been designed and tested. It is fully functional.

In preparation for constructing large sensor arrays with hundreds of elements, the possibility of constructing defect tolerant tactile cells was explored. Analyses based on the Poisson model indicate that working arrays with 1,000 functional cells are possible if computing elements are replicated within each tactile cell. Experiments on a 3x3 array sensor with redundant pairs of computing elements suggest that large tactile sensing arrays are within reach.

[Song 81a]

Song, S.W.

On a High-Performance VLSI Solution to Database Problems.

PhD thesis, Carnegie-Mellon University, July, 1981.

Also available as a CMU Computer Science Department technical report, VLSI Document V075, August 1981.

This thesis explores the design and use of custom-made VLSI hardware in the area of database problems. Our effort differs from most previous ones in that we search for structures and algorithms, directly implementable on silicon, for the solution of computation-intensive database problems. The types of target database systems include the general database management systems and the design database systems. The thesis deals mainly with database systems of the relational model. One common view concerning special-purpose hardware usage is that it performs a specific task. The proposed device is not a hardware solution to a specific problem, but provides a number of useful data structures and basic operations. It can be used to improve the performance of any sequential algorithm which makes extensive use of such data structures and basic operations. The design is based on a few cells, interconnected in the form of a complete binary tree. The proposed device

can handle all the basic relational operations: select, join, project, union, and intersection.

- [Song 81b] Song, S.W.
I/O Complexity and Design of Special-Purpose Hardware for Sorting.
 Technical Report VLSI Document V075, Carnegie Mellon University Computer Science Department,
 February, 1981.
 For abstract, see Song's PhD thesis "On a High-Performance VLSI Solution to Database Problems," 1981.
- [Sproull 81a] Sproull, R.F.
Using Program Transformations to Derive Line-Drawing Algorithms.
 Technical Report CMU-CS-81-117, Carnegie Mellon University Computer Science Department,
 April, 1981.
- [Sproull 81b] Sproull, R. F.
 Simple Color Checkpoints.
*VLSI Design*2(2), August, 1981.
- [Sproull 81c] Sproull, R.F., I.E. Sutherland, A. Thompson, S. Gupta, and C. Minter.
The 8 x 8 Display.
 Technical Report CMU-CS-82-105, Carnegie Mellon University Computer Science Department,
 December, 1981.
- [Steele 81] Steele, G.L.
 VLSI Systems and Computations.
 In *Proceedings of the 1981 Conference on VLSI Systems and Computations*, Carnegie-Mellon University, October, 1981.
- [Sugie 84] Sugie, M., O. Menziloglu, and H.T. Kung.
 CARGuide—On Board Computer for Automobile Route Guidance.
 In *Proceedings of the 1984 National Computer Conference*, Pages 695-706. July, 1984.
- [Tanner 81] Tanner, J. E., M.H. Raibert, and R. Eskenazi.
 A VLSI Tactile Sensing Array Computer.
 In *Proceedings of the 1981 CalTech Conference on VLSI Systems*, California Institute of Technology, February, 1981.
- [Tsao 81] Tsao, M.M., A.W. Wilson, R.C. McGarity, C.J. Tseng, and D.P. Siewiorek.
 C.fast: A Fault Tolerant and Self Testing Microprocessor.
 In *VLSI Systems and Computations*, Pages 357-366. Computer Science Press, Inc., October, 1981.
- [Tsao 82] Tsao, M., A. Wilson, R. McGarity, C.J. Tseng, and D.P. Siewiorek.
 Design of a C.fast: A Single Chip Fault-Tolerant Microprocessor.
 In *Proceedings of the 12th Fault Tolerant Computing Symposium (FTCS-12)*, IEEE Computer Society, June, 1982.

- [Walker 81] Walker, H.
A Time-Multiplexed Crossbar for the NETL Machine?
 Technical Report VLSI Document V094, Carnegie Mellon University Computer Science Department,
 1981.
- [Walker 82] Walker, H.
Yield Simulation for Integrated Circuits - A Thesis Proposal.
 Technical Report VLSI Document V121, Carnegie Mellon University Computer Science Department,
 October, 1982.
- [Walker 83a] Walker, H. and S. Director.
 Yield Simulation for Integrated Circuits.
 In *ICCAD83*, Pages 256-257, September, 1983.
- [Walker 83b] Walker, H.
The Control Store and Register File Design Of The Programmable Systolic Chip.
 Technical Report CMU-CS-83-133, Carnegie Mellon University Computer Science Department,
 May, 1983.
 The design of a 64 word by 60-bit writable control store and a 64 word by 9-bit register file is described. A dynamic three-transistor NMOS RAM cell is used to meet area constraints. Sense amplifiers reduce access time and bootstrapped logic reduces power dissipation. A shift register is used to load the control store. Experimental results indicate a typical read/write cycle time of 160ns and power dissipation of 150mW for the control store.
- [Walker 83c] Walker, H.
4-KBIT Four-Transistor Dynamic RAM.
 Technical Report CMU-CS-83-140, Carnegie Mellon University Computer Science Department,
 June, 1983.
- [Weiser 81] Weiser, U. and Davis, A.
 A Wavefront Notation Tool for VLSI Array Design,
 In Kung, H.T., Sproull, R.F., and Steele, G.L., Jr., *VLSI Systems and Computations*, Pages 226-234. Computer Science Department, CMU, Computer Science Press, Inc., 1981.
 This paper presents an overview of an extension to a mathematically based methodology for mapping an algorithmic description into a concurrent implementation on silicon. The result of this methodology yields a systolic array. The basic mathematical method was initially described by Cohen. Extensions were made by Weiser and Davis, and a number of others. This approach focuses on the correspondence between equations defining a certain computation and networks which perform the computation. As the complexity of problems increases, a hierarchical approach can reduce the complexity by hiding detail and thus reduce the design complexity at each level. The purpose of this paper is to introduce a method for treating sets of data as wavefront entities in the equations of the mathematical methodology and in the graphical representation.

Index

Accent 6-1
 Active database 5-7
 Ada 5-1
 Ada+ 5-3
 Advanced environments 5-6
 Aerial images 3-5
 Aerial mapping 3-7
 Analogy 4-2
 Atomic transactions 5-4
 Authentication 5-7
 Automated image analysis 3-6

 B* search algorithm 4-3
 Beam search 8-4
 BKG 4-3
 Boltzmann 4-4

 Camera motion 3-8
 Cartography 3-7, 3-9
 Chess machine 8-5
 Chunking 4-3, 4-6
 Circuit extractors 8-6
 Cm* 5-4
 Coarse-class lattice 4-10
 Code optimization 5-2
 Communication costs 8-3
 Communication protocols 6-2
 Compiler generator 5-3
 Compiler-generator 5-1
 Compiling techniques 5-3
 Cooperative User Interface 7-1
 Coordinate transformations 3-6
 Correctness 5-5, 8-2
 Cousin-Spice 7-1, 7-2
 Cousin-Unix 7-1, 7-2

 Data acquisition system 3-6
 Database processing 8-4
 Depth information 3-3
 Derivational analogy 4-2
 Designer 4-7
 Diana 5-2
 Digital mapping 3-6
 Distributed sensor network 6-1
 DPL-82 6-4
 DSN 6-1
 DYPAR 7-3

 Fault recovery 6-2, 6-3
 Fault-tolerant systems 5-5
 Feature 4-9
 Fido 3-8
 Form-Editor 7-4
 Form-Manager 7-4

 Gandalf 5-3
 Gandalf system 5-7

Generalized cylinders 3-2
Generators 5-2
Generic compiler-generating 5-1
Gnome 5-8
Gradient 3-1
Gradient space theory 3-1

Hitech 4-8

Iago 4-3
IDL 5-2
Illumination surfaces 3-2
Image matching 3-3
Image/map database 3-6
Inheritance 4-4
Interconnection chip 8-5
Interest operators 3-8
Interface specification 6-3
Interprocess communication 6-1

Language analysis 5-1
Learning 4-6, 4-8
LINC 8-5
Load-balancing 6-3

Machine learning 4-2
MAPS 3-7, 4-10
Matchmaker 6-3
Method of differences 3-8
Model-based vision 3-5
Modeling 3-4
Multi-view images 3-5
Multiprocessors 5-4

Navigation 3-7
NETL 4-4

Occluding contours 3-2, 3-5
Operating system 6-1
Optical flow 3-8
Optimizing compilers 5-3

Parallel architectures 5-4
Path relaxation 3-8
Pattern recognition algorithms 8-5
Phoenix 3-9
Phonetic lattice 4-10
Photointerpretation 3-6
PQCC 5-2
Problem decomposition 8-3
Prodigy 4-8
Production Quality Compiler Compiler 5-2
Program library 3-6
Program transformation 8-2
Programmable architectures 8-3
Programmable symbolic chip 8-5
Programming-in-the-large 5-6
Programming-in-the-small 5-6
Project management 5-7
PSC 8-5

INDEX

QBKG 4-3

R1-Soar 4-6
Range images 3-5
Rangefinder 3-5
Reconfiguration 5-5
Redundancy 5-4
Regular expression matching 8-5
Reliability 5-4, 6-4
Remote procedure call 6-3
Replication 5-4
Rover 3-8
Runtime support 5-2

Sapphire 7-5
Scanline 3-3
Security 5-4
Shadow geometry theory 3-2
Shape inference 3-2
Shape matching 3-6
Soar 4-5
Solids of revolution 3-2
SPAM 4-10
Speech recognition 4-9
Spice 2-1
SRI testbed 3-9
Stardust 6-3
Stereo 3-3, 3-9
Superpuzz 4-4
Synchronization 8-3
System building blocks 8-3
System reliability 5-3
System security 5-3
Systolic algorithms 8-2

TABS 5-5, 6-4
Tactile sensor chips 8-5
Target programming environment 5-6
Task-specific architectures 8-3
Temporal logic 5-5
Testing custom chips 8-6
Transaction 6-4
Transformational analogy 4-2

Universal subgoalng 4-5

Velocity fields 3-9
Verification 5-5
Virtual memory 6-1
Voice message system 4-9, 7-6

Wafer scale integration 8-3
Warp 8-3
Warp processor 8-5
Weak methods 4-5
World modellers 4-2

END

12-86

DTIC